Projection Algebra Analysis of Error Correcting Codes

Jonathan S. Yedidia
MERL
201 Broadway, 8th Floor
Cambridge, MA 02139
yedidia@merl.com

Erik B. Sudderth
LIDS
Dept. of EECS, MIT
Cambridge, MA 02139
esuddert@mit.edu

Jean-Philippe Bouchaud
SPEC CEA-Saclay
Orme des Merisiers
91191 Gif sur Yvette, France
bouchaud@spec.saclay.cea.fr

Abstract

We explain the projection algebra technique, which makes possible an exact computation of the performance of an arbitrary parity-check error-correcting code as decoded by the belief propagation algorithm for the binary erasure channel. This technique improves on density evolution by exactly accounting for the statistical dependencies that exist between belief propagation messages. Although the exact projection algebra technique is computationally intractable for codes of large block-length, it can be efficiently approximated to give rigorous upper and lower bounds on the bit error rates of arbitrary parity-check codes.

1 Introduction

We consider the problem of efficiently computing the bit error rate of an arbitrary block parity-check code, as decoded by a belief propagation (BP) decoder and used over the binary erasure channel (BEC). Our motivation is that the results of such a computation can serve as an objective function in a search for good codes. The pioneering paper by Luby, et. al. [1], which analyzed and optimized irregular low-density parity-check (LDPC) codes decoded by a BP decoder and used over the BEC, demonstrated that one could design capacity-approaching codes by optimizing the codes for their performance as measured using the density evolution technique. The name "density evolution" was in fact introduced in the important paper by Richardson and Urbanke [2], who extended the technique to handle more general decoders and channels.

Our own "projection algebra" technique builds on the density evolution technique, and ameliorates its major flaw. Namely, density evolution only becomes exact when the Tanner graph representing the code is cycle-free, while practical codes never have cycle-free graphs. Of course, density evolution is still useful, because for some classes of codes, one can demonstrate that as the block-length N approaches infinity, the effects of cycles can be neglected. However, that has meant that analyses using the density evolution technique have been effectively limited to appropriate classes of codes in the infinite block-length limit.

In contrast, our technique applies to any block parity-check code of arbitrary block-length. Recently, Di, et. al. [3] have shown how to perform a finite length analysis of regular LDPC codes used over the BEC. They were able to analyze both BP decoding and maximum likelihood decoding, but their analyses was limited to averages over ensembles of regular LDPC codes. Our analysis as presented here also only applies to the BEC, and we are further limited to BP decoding, but the advantage of our technique is that we are able to analyze the performance of any given *arbitrary* parity-check code.

Two of the present authors recently wrote a paper on a "renormalization group" (RG) technique with similar motivations [4]. That paper also contains some additional background material which may be helpful to some readers. The relative advantages of the projection algebra technique are that it is exact for codes of small block-length, and that for larger codes, we can efficiently obtain rigorous and rather tight bounds on the bit error rate. In the RG approach, on the other hand, the approximations are comparatively poorly controlled.

2 Projection Algebra

We begin by developing the rather straightforward mathematics of projection algebra, as motivated by the context of our problem of analyzing BP decoding of a parity-check code used over the BEC.

2.1 Definitions

Because parity-check codes are linear and the BEC is symmetric, we can assume, for the purpose of analysis, that the all-zeros block is always transmitted. That means that after passing through the channel, the block is received as a series of N bits that are all `0` or `?`, where we use the symbol `?` to denote an erasure. We will assume throughout that the channel is memoryless, so that the N bits will be erased independently.

We index the N bits with the letter i and take the erasure probability for the ith bit to be x_i. Normally, of course, the erasure probability of every bit should be equal, but it will prove very useful to consider them separately. As an example, if $N = 3$, then $2^3 = 8$ possible blocks could be received, and the block `0?0` will be received with probability $(1 - x_1)x_2(1 - x_3)$. More generally, we will focus on the probabilistic model such that the sample space is the set of 2^N possible received blocks, and a given received block R is received with probability

$$p(R) = \prod_{i \in R_?} x_i \prod_{j \in R_0} (1 - x_j) \tag{1}$$

where $R_?$ is the set of bits erased by the channel and R_0 is the set of correctly received bits.

We will be interested in computing quantities that are averages over all possible received blocks, as weighted by the probabilities of the received blocks. Thus, we will be interested in "events" E that are formally defined as sets of possible received blocks. The probability of an event is the sum of the probabilities of the received blocks in the set. For example, if $N = 3$, the event E that the first and second bit are erased consists of the set of received blocks $\{??0, ???\}$ and has probability $p(E) = x_1x_2(1-x_3)+x_1x_2x_3 = x_1x_2$.

We introduce the notion of a "projected polynomial," as a function of the variables x_i. Projected polynomials are defined to be the subspace of ordinary polynomials such that no term in a projected polynomial has a non-linear dependence on any of the x_i. For example, $x_1x_2 - 2x_3$ is a projected polynomial, but $x_1x_2 - 2x_3^2$ is not a projected polynomial because of the quadratic dependence on x_3. Notice from equation (1) that the probability of any possible received block is a projected polynomial as a function of the x_i. Since the sum of any two projected polynomials is also a projected polynomial, the probability of any event in our probabilistic model must also be given by a projected polynomial. Of course, the probability of an event must also satisfy some more conditions: namely, it must evaluate to a real number between zero and one when all the x_i are set to real numbers between zero and one.

The ordinary product of two projected polynomials might not be a projected polynomial. We define the operation of "projective multiplication" $p_1 \otimes p_2$ of two projected polynomials p_1 and p_2 to be the same as ordinary multiplication, except that any exponents greater than one in the ordinary product are simply reduced to one. For example, if $p_1 = x_1 x_2 + x_3$, and $p_2 = x_1 + x_2 x_3$, then the ordinary product would be $p_1 p_2 = x_1^2 x_2 + x_1 x_2^2 x_3 + x_1 x_3 + x_2^2 x_3$, while the projective product would be $p_1 \otimes p_2 = x_1 x_2 + x_1 x_2 x_3 + x_1 x_3 + x_2 x_3$. Projective multiplication is "projective" in the sense that it "projects" an ordinary polynomial into the subspace of projected polynomials.

It is easy to verify that projective polynomials are a mathematical field over the operations of ordinary addition and projective multiplication. That means that the standard commutative, associative, and distributive laws may be used. We refer to this field as "projection algebra."

2.2 Projection Algebra as a Calculus of Probabilities

We note, using equation (1) and the definition of projective multiplication, that for any possible received block R, $p(R) \otimes p(R) = p(R)$, while for any two distinct received blocks R_1 and R_2, $p(R_1) \otimes p(R_2) = 0$. From this, we obtain the following important result: for any two events E_1 and E_2, the probability of the intersection of E_1 and E_2 is given by

$$p(E_1 \cap E_2) = p(E_1) \otimes p(E_2). \tag{2}$$

We can prove this by expanding $p(E_1)$ and $p(E_2)$ into the sums of the probabilities of their constituent received blocks. The only surviving cross-terms in $p(E_1) \otimes p(E_2)$ will correspond to those received blocks that are in $E_1 \cap E_2$.

Note that one normally has $p(E_1 \cap E_2) = p(E_1)p(E_2)$ only if the events E_1 and E_2 are statistically independent. The result of equation (2) holds for any two events, even if they are statistically dependent.

We also obtain a result for the union of E_1 and E_2:

$$p(E_1 \cup E_2) = p(E_1) + p(E_2) - p(E_1) \otimes p(E_2) \tag{3}$$

using equation (2) and the general laws of probability. More generally, we find that for n distinct events E_i,

$$p(\bigcap_{i=1}^{n} E_i) = \bigotimes_{i=1}^{n} p(E_i) \tag{4}$$

and

$$p(\bigcup_{i=1}^{n} E_i) = 1 - \bigotimes_{i=1}^{n} (1 - p(E_i)) \tag{5}$$

where $\otimes$ is used to represent a projective product.

3 Application to BP Decoding for the BEC

We presume that the reader is familiar with the parity-check matrix and corresponding Tanner graph representations of parity-check codes [4]. The "variable nodes" in a Tanner graph correspond to the N bits transmitted over the channel, while the "check nodes" correspond to the M parity checks given by the M rows of an M by N parity check matrix. We label the variable nodes with indices beginning with the letter i, and label the check nodes with indices beginning with the letter a.

We review the operation of the BP decoding algorithm for the BEC. The BP decoding algorithm works in discrete time iterations, which we label with the integer t. There are messages from variable nodes to connected check nodes $m_{ia}[t]$ and messages from check nodes to connected variable nodes $m_{ai}[t]$. All the messages can be in one of three states: `0`, `1` or an erasure `?`. We initialize the messages $m_{ai}[t=0]$ to be erasures, and then update the $m_{ia}[t]$ and $m_{ai}[t]$ messages in turn at each iteration, as follows.

At each iteration $t \geq 1$, a message $m_{ia}[t]$ will be an erasure if all incoming messages $m_{bi}[t-1]$ from other check nodes and from the channel are also erasures. Otherwise it will take on the value of any incoming non-erasure messages, as such messages will always be reliable. All check nodes will in turn send a message $m_{ai}[t]$ to each connected variable node. Such a message will be an erasure if an erasure is received from any other variable node involved in the constraint. Otherwise, it will be the binary sum of the incoming messages $m_{ja}[t]$. A variable node is successfully decoded at the end of an iteration if any check node or the channel sends it a non-erasure message.

Recall that without loss of generality, we can assume that the all-zeros block is transmitted, so we can safely ignore the possibility that any `1` messages are ever sent in the decoding algorithm. Thus we can completely characterize the functioning of the BP decoding algorithm by focusing on the following events: $F_{ia}[t]$, which represents the event that variable node i sends check node a an erasure at iteration t; $G_{ai}[t]$, which is the event that check node a sends variable node i an erasure at iteration t; X_i, which is the event that variable node i is erased by the channel; and $B_i[t]$, which is the event that variable node i fails to be successfully decoded at iteration t. These events are related by the following equations, which transcribe the functioning of the algorithm:

$$F_{ia}[t] = X_i \cap \bigcap_{b \in N(i) \backslash a} G_{bi}[t-1], \tag{6}$$

$$G_{ai}[t] = \bigcup_{j \in N(a) \backslash i} F_{ja}[t] \tag{7}$$

$$B_i[t] = X_i \cap \bigcap_{a \in N(i)} G_{ai}[t] \tag{8}$$

where $N(a)\backslash i$ denotes the set of nodes that are connected to node a excluding node i.

We are interested in computing probabilities of these events averaged over all possible received blocks. We can do this by using a projected polynomial to represent the probability of each event of interest, and then using the above equations, combined with equations (4) and (5), to relate the probabilities. We find the iterative equations:

$$p(F_{ia}[t]) = x_i \otimes \bigotimes_{b \in N(i) \backslash a} p(G_{bi}[t-1]) \tag{9}$$

$$p(G_{ai}[t]) = 1 - \bigotimes_{j \in N(a) \backslash i} (1 - p(F_{ja}[t])) \tag{10}$$

$$p(B_i[t]) = x_i \otimes \bigotimes_{a \in N(i)} p(G_{ai}[t]). \tag{11}$$

$p(G_{ai}[t=0]) = 1$ is the initialization condition. $p(B_i[t \to \infty])$ is a quantity of particular interest, as it represents the bit error rate at node i after the decoding has completed.

Notice that these equations are identical to the density evolution equations that would be derived by assuming all incoming messages to a node are statistically independent, except that probabilities are now represented as projected polynomials instead of real numbers, and projective products are used instead of ordinary products. We emphasize that these changes result in the above equations being *exact*, even when the probabilities are statistically dependent.

4 The Critical Representation

We can exploit some properties of the events we are interested in, and the BEC, to represent them more compactly. In particular, the events $F_{ia}[t]$, $G_{ai}[t]$, X_i, and $B_i[t]$ all have the property that they can be expressed as unions of *critical events*: for any such event E, we can write $E = \bigcup_k C_k$, where the C_k are its critical events.

A critical event is an event consisting of the subset of all possible received blocks for which a certain set of nodes (the *critical set*) are erased by the channel. For example, an event defined by the fact that nodes 1, 2, and 4 are erased is a critical event, and the nodes $1, 2, 4$ would be the corresponding critical set. We denote a critical event by simply listing the nodes in its critical set in angled brackets. The probability of a critical event is simply the product of the x_i corresponding to the nodes in the critical set; for our example, $p(\langle 1, 2, 4\rangle) = x_1 x_2 x_4$.

The reason that the events we are interested in can be expressed as unions of critical events is that additional erasures by the channel always strictly impair the decoding process. Thus, it cannot be that erasing a critical set of nodes would cause an erasure, but erasing some additional node or nodes would somehow fix the problem. This means that we can always break down a "problem" event, such as an erasure occuring at a particular message, into the minimal sets of erasures that can cause that problem.

The probability of an event represented by its critical sets can be computed straightforwardly using equation (5). For example, if $E = \langle 1, 2\rangle \cup \langle 2, 3\rangle \cup \langle 4\rangle$, then

$$\begin{aligned} p(E) &= 1 - (1 - x_1 x_2) \otimes (1 - x_2 x_3) \otimes (1 - x_4) & (12) \\ &= x_1 x_2 + x_2 x_3 + x_4 - x_1 x_2 x_3 - x_1 x_2 x_4 - x_2 x_3 x_4 + x_1 x_2 x_3 x_4. & (13) \end{aligned}$$

We notice that the critical representation of an event is more compact than the corresponding projected polynomial for its probability. We can avoid unnecessary conversions into projected polynomials by noting that the union and intersection of critical events are also critical events, and defining intersection and union operations directly. The intersection of two critical events is just the critical event with a critical set given by the *union* of the nodes in the two critical sets; e.g. $\langle 1, 2\rangle \cap \langle 2, 3\rangle = \langle 1, 2, 3\rangle$. If we have two events that are expressed in terms of their critical events as $E_1 = \cup_i C_i$ and $E_2 = \cup_j D_j$, then

$$E_1 \cap E_2 = \bigcup_{(i,j)} (C_i \cap D_j) \qquad (14)$$

where (i, j) represents all pairs of indices i and j. The result can be simplified by removing any redundant critical events (events whose critical set is a superset of some other included critical set.) For example, if $E_1 = \langle 1, 2\rangle \cup \langle 2, 3\rangle$, and $E_2 = \langle 1, 2, 3\rangle \cup \langle 3, 4\rangle$, then

$$\begin{aligned} E_1 \cap E_2 &= \langle 1,2,3\rangle \cup \langle 1,2,3,4\rangle \cup \langle 1,2,3\rangle \cup \langle 2,3,4\rangle \\ &= \langle 1,2,3\rangle \cup \langle 2,3,4\rangle \end{aligned} \quad (15)$$

We also obtain

$$E_1 \cup E_2 = \bigcup_i C_i \cup \bigcup_j D_j. \quad (16)$$

which can again be simplified by removing any redundant critical events. For example, with E_1 and E_2 as given above, we find $E_1 \cup E_2 = \langle 1,2\rangle \cup \langle 2,3\rangle \cup \langle 3,4\rangle$, where we remove the critical event $\langle 1,2,3\rangle$ because it is contained in other critical events.

We can track the evolution of the BP algorithm, as given by equations (6, 7, 8), by representing each event in its critical representation and performing the necessary unions and intersections as described above. The conversion into projected polynomials and the final numerical evaluation of probabilities can be postponed until it is actually needed. Besides the advantage of compactness, the critical representation gives us important direct information about the patterns of erasures that will cause decoding failures.

5 A Small Example

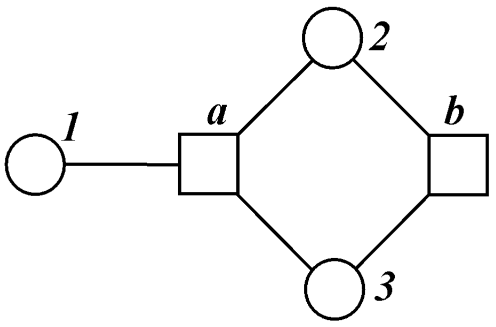

Figure 1: Tanner graph for a parity-check code with three variable nodes, labeled 1, 2, and 3, and two check nodes, labeled a and b.

Consider the tiny code represented by the Tanner graph in Figure 1. We track the events $F_{1a}[t]$, $F_{2a}[t]$, $F_{2b}[t]$, $F_{3a}[t]$, $F_{3b}[t]$, $G_{a1}[t]$, $G_{a2}[t]$, $G_{a3}[t]$, $G_{b2}[t]$, $G_{b3}[t]$, $B_1[t]$, $B_2[t]$, and $B_3[t]$. By definition, the events X_i are fixed at $X_1 = \langle 1\rangle$, $X_2 = \langle 2\rangle$, and $X_3 = \langle 3\rangle$. The events $G_{a1}[0]$, $G_{a2}[0]$, $G_{a3}[0]$, $G_{b2}[0]$, and $G_{b3}[0]$ are all initialized to be certain events, which means that they are represented by the empty critical set $\langle\rangle$.

We iterate equations (6), (7), and (8), starting with equation (6). We find

$$F_{1a}[1] = \langle 1\rangle; F_{2a}[1] = \langle 2\rangle; F_{2b}[1] = \langle 2\rangle; F_{3a}[1] = \langle 3\rangle; F_{3b}[1] = \langle 3\rangle. \quad (17)$$

Then, using equation (7), we find

$$G_{a1}[1] = \langle 2\rangle \cup \langle 3\rangle; G_{a2}[1] = \langle 1\rangle \cup \langle 3\rangle; G_{a3}[1] = \langle 1\rangle \cup \langle 2\rangle; G_{b2}[1] = \langle 3\rangle; G_{b3}[1] = \langle 2\rangle. \quad (18)$$

Using equation (8), we find

$$B_1[1] = \langle 1,2\rangle \cup \langle 1,3\rangle; B_2[1] = \langle 2,3\rangle; B_3[1] = \langle 2,3\rangle. \quad (19)$$

Let us examine just one of these computations–the one for $B_2[1]$–in more detail. We have

$$B_2[1] = X_2 \cap G_{a2}[1] \cap G_{b2}[1], \tag{20}$$

which can be interpreted as "variable node 2 will not be decoded on the first iteration if it is erased by the channel and both check nodes send it an erasure message." Using our previous results, we find

$$B_2[1] = \langle 2 \rangle \cap (\langle 1 \rangle \cup \langle 3 \rangle) \cap \langle 3 \rangle, \tag{21}$$

which can be interpreted as "variable node 2 will not be decoded on the first iteration if the channel erases node 2, and the channel erases node 1 or node 3, and the channel erases node 3." Using our rules for manipulating critical representations, which are really just rules of logic, we find

$$\begin{aligned} B_2[1] &= (\langle 1,2 \rangle \cup \langle 2,3 \rangle) \cap \langle 3 \rangle \\ &= \langle 1,2,3 \rangle \cup \langle 2,3 \rangle \\ &= \langle 2,3 \rangle. \end{aligned} \tag{22}$$

which can be interpreted as "variable node 2 will not be decoded on the first iteration if the channel erases nodes 2 and 3."

Continuing to solve the iterative equations, we find

$$\begin{aligned} F_{1a}[t \geq 2] = \langle 1 \rangle;\ F_{2a}[t \geq 2] &= \langle 2,3 \rangle;\ F_{2b}[t \geq 2] = \langle 1,2 \rangle \cup \langle 2,3 \rangle; \\ F_{3a}[t \geq 2] &= \langle 2,3 \rangle;\ F_{3b}[t \geq 2] = \langle 1,3 \rangle \cup \langle 2,3 \rangle; \end{aligned} \tag{23}$$

$$\begin{aligned} G_{a1}[t \geq 2] = \langle 2,3 \rangle;\ G_{a2}[t \geq 2] &= \langle 1 \rangle \cup \langle 2,3 \rangle; \quad G_{a3}[t \geq 2] = \langle 1 \rangle \cup \langle 2,3 \rangle; \\ G_{b2}[t \geq 2] &= \langle 1,3 \rangle \cup \langle 2,3 \rangle;\ G_{b3}[t \geq 2] = \langle 1,2 \rangle \cup \langle 2,3 \rangle; \end{aligned} \tag{24}$$

and

$$B_1[t \geq 2] = \langle 1,2,3 \rangle;\ B_2[t \geq 2] = \langle 2,3 \rangle;\ B_3[t \geq 2] = \langle 2,3 \rangle. \tag{25}$$

We can easily evaluate the probabilities of any of these events. As one example, we find, using equation (3),

$$\begin{aligned} p(G_{b2}[t \geq 2]) &= p(\langle 1,3 \rangle) + p(\langle 2,3 \rangle) - p(\langle 1,3 \rangle) \otimes p(\langle 2,3 \rangle) \\ &= x_1 x_3 + x_2 x_3 - x_1 x_3 \otimes x_2 x_3 \\ &= x_1 x_3 + x_2 x_3 - x_1 x_2 x_3. \end{aligned} \tag{26}$$

A few comments are in order. First, we may of course set $x_1 = x_2 = x_3 = x$ and read off results for the ordinary case when all the erasure probabilities are equal. We note from equations (25) that the bit error rates differ from bit to bit, and that the average bit error rate is $(2x^2 + x^3)/3$. However, our technique gives considerably more information than just the average bit error rates–we actually find probabilities for every possible erasure message at every iteration, including the explicit error patterns that will cause erasures. All of the results in this example can be verified by explicitly considering the eight possible received blocks and summing over them, tracking how BP decoding would proceed for each one of them.

6 Lower and Upper Bounds

As the blocklength N increases, the number of possible received blocks grows as 2^N, so exact calculations performed by summing over all possible received blocks become intractable. Unfortunately, the exact projection algebra method also becomes intractable, even when using the critical representation, because the number of critical sets needed to represent an event will also ultimately grow exponentially in N. Nevertheless, the projection algebra method is of interest even for large N, as it provides a basis for simple and tractable approximations that will give rigorous lower and upper bounds on the probabilities of all events of interest.

6.1 Lower Bounds

Recall that we can represent all the events of interest to us as unions of critical events: $E = \bigcup_k C_k$. We introduce the approximation of taking any of the events of interest, and simply removing one or more of the critical events C_k from its union. We first prove that this approximation gives rigorous lower bounds on the probabilities of all events of interest.

Proof: If we take one such event E, and eliminate one or more of the critical events from the union, we obtain another event $E_<$ that is a strictly smaller set of possible received blocks. Thus, the event $E_<$ is a subset of event E, and $p(E_<) < p(E)$. If we further consider such subsets of two or more events E_i, then the intersection and union of the subsets must be subsets of the intersection and union of the original events E_i. Since equations (6), (7), and (8) for the analysis of BP decoding in the BEC all contain just unions and intersections on their right-hand sides, the approximated events on the left hand side must also be subsets of their corresponding events, and their probabilities will consistently be lower bounds on the true probabilities of the events on the left hand sides. *Q.E.D.*

There are many ways to implement such lower bounds. In practice, we store the critical representation of an event as a list of critical sets. One can keep thousands of critical sets for each event without exceeding the memory limitations of modern-day computers. If one must remove critical sets, or simply desires to do so to obtain results in less time, it is appropriate to remove the *largest* critical sets, as these will make the smallest contribution in the $x \to 0$ limit which is of greatest interest to us. (The size L of a critical set is defined as the number of nodes in the critical set.) We have found that a good heuristic is to systematically eliminate, for each event E of interest, all critical sets that have a size $L > L_{min}(E) + \theta$, where $L_{min}(E)$ is the size of the smallest critical set for event E, and θ is a positive parameter that defines the approximation. Larger values for θ will give tighter lower bounds but require more memory and computation time.

6.2 Upper Bounds

To obtain an upper bound on an event $E = \bigcup_k C_k$ in its critical representation, we take any two of its critical events, say C and D, and replace them with the critical event V that has a critical set which is the *intersection* of the two critical sets; e.g. if $C = \langle 1, 2\rangle$ and $D = \langle 2, 3\rangle$, then $V = \langle 2\rangle$. It is easy to convince oneself that V must be a superset of $C \cup D$. For the example given, if $N = 3$, then $C = \{\texttt{??0}, \texttt{???}\}$, $D = \{\texttt{0??}, \texttt{???}\}$, and $V = \{\texttt{0?0}, \texttt{0??}, \texttt{??0}, \texttt{???}\}$, which is a superset of $C \cup D = \{\texttt{0??}, \texttt{??0}, \texttt{???}\}$. Thus, making this replacement will give an upper bound on $p(E)$. By an argument identical to that given above for the lower bounds, we can prove that if we only make such replacements, we will consistently obtain supersets for all events of interest as we analyze BP decoding for the BEC, and upper bounds on all their probabilities.

Again there are many possible ways to implement such upper bounds. We have found that the following is a good procedure for obtaining upper bounds. One begins by sorting the list of critical sets representing a given event E by their length. Then, beginning with the longest critical sets, one tries to find pairs of critical sets such that their intersection is larger than $L_{min}(E) + \phi$, where again $L_{min}(E)$ is the smallest critical set for event E, and ϕ is a parameter defining the approximation. If such a pair is found, the critical sets are replaced by their intersection, and any newly redundant critical sets (which are now supersets of the intersection) are also removed. The procedure continues until no good candidate pairs are found. Unfortunately, computing upper bounds in this way is considerably slower than computing lower bounds as described above, because searching for discardable pairs of critical sets is much slower than simply checking the sizes of critical sets. Again, ϕ is a parameter which lets one trade off the tightness of the bound against the expense in memory and computation time.

6.3 Miscellaneous Implementation Issues

A couple of implementation issues should be mentioned. First, when a computation requires numerous unions or intersections of events to be taken, and we are bounding events as above, we always systematically bound the intermediate results, to keep the size of all computations limited. This has the drawback that our bounds on probabilities will ultimately depend slightly on the order in which we take unions or intersections of events.

Second, the final phase of actually converting events into their probabilities may be slow in practice because one needs to fully expand the critical representation into a potentially huge projected polynomial. However, one can obtain a good approximation to the probability of an event $E = \bigcup_k C_k$, which strictly speaking is given by $p(E) = 1 - \bigotimes_k (1 - p(C_k))$, by computing $p(E) \approx 1 - \prod_k (1 - p(C_k))$. This approximation can be computed very quickly because one can numerically evaluate all the $p(C_k)$ before taking the products, and it captures the leading behavior in the $x \to 0$ limit, assuming that redundant critical events are always removed.

7 Results for a regular LDPC Code

We have verified, for many small ($N \leq 20$) randomly chosen parity check codes, that the bounds described in the previous section will become tight for large enough θ and ϕ. Usually, $\theta \geq 2$ and $\phi \geq 2$ are sufficient to obtain excellent bounds.

For N large enough that exact bit error rates become impractical to compute, we are still able to compute lower bounds that appear to be tight for small erasure or bit error rates. In Figure 2, we show such lower bounds for a randomly chosen $N = 50$ regular $(3, 5)$ LDPC code. We computed lower bounds using $\theta = 0, 1, 2$, and compare with simulations. Of course, the lower bounds have the advantage over simulations that they can be efficiently computed even at arbitrarily small erasure rates.

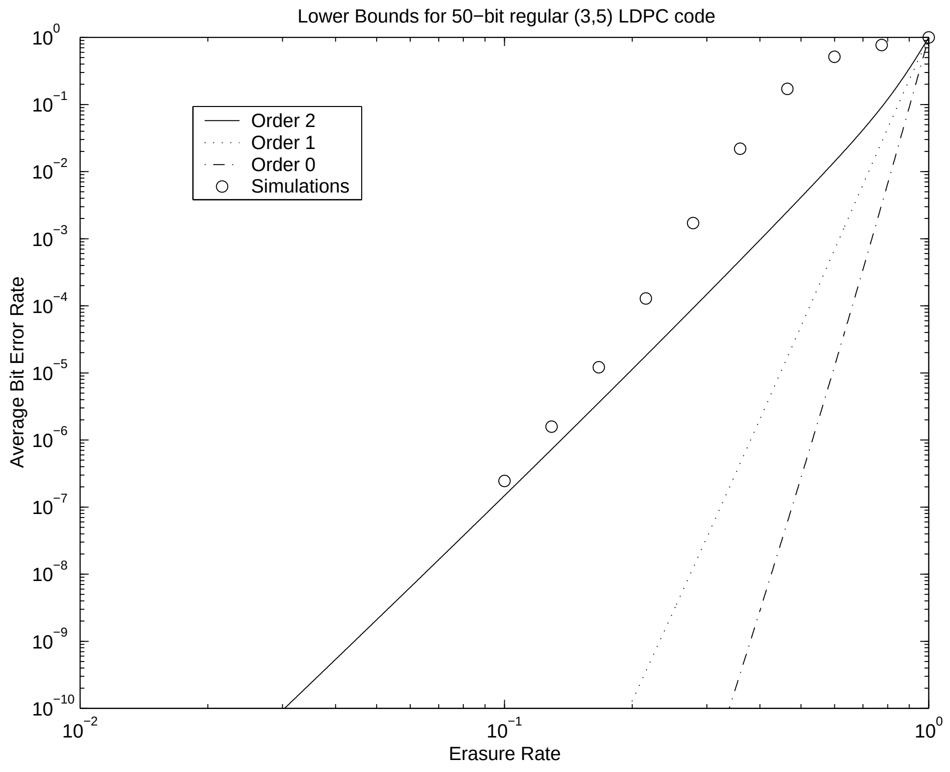

Figure 2: Lower bounds compared with simulations for a regular LDPC code. The error bars on the simulations are smaller than the symbol sizes. Note that the $\theta = 2$ bound appears to be tight in the limit of small erasure or bit error rates.

8 Outlook

The techniques described here can be extended to analyze the performance of parity-check codes decoded by message-passing decoding algorithms on the binary symmetric channel. We are currently working on this problem, and we also hope to use our analysis tools as a basis for optimizing finite blocklength codes under BP decoding.

References

[1] M.G. Luby, M. Mitzenmacher, M.A. Shokrollahi, and D.A. Spielman, "Improved Low-Density Parity-Check Codes Using Irregular Graphs," *IEEE Trans. on Information Theory*, vol. 47, no. 2, pp. 585-598, February 2001.

[2] T.J. Richardson and R.L. Urbanke, "The Capacity of Low-Density Parity-Check Codes Under Message-Passing Decoding," *IEEE Trans. on Information Theory*, vol. 47, no. 2, pp. 599-617, February 2001.

[3] C. Di, D. Proietti, E. Telatar, T.J. Richaradson, and R.L. Urbanke, "Finite Length Analysis of Low-Density Parity-Check Codes," available online at http://lthcwww.epfl.ch/publications.html, 2001.

[4] J.S. Yedidia and J.-P. Bouchaud, "Renormalization-Group Approach to Error Correcting Codes," available online at http://www.merl.com/papers/TR2001-19/, 2001.