Bibliography

[ASH2006] SHOKROLLAHI, M. A.. **Raptor codes**. IEEE Trans. Inform. Theory, 52(6):2551–2567, June 2006.

[BEB1998] BENEDETTO, S.; BIGLIERI, E.. **Principles of Digital Transmission: With Wireless Applications**. Plenum Publishers Corp, 1998.

[CBG1996] BERROU, C.; GLAVIEUX, A.. **Near optimum error correcting coding and decoding: turbo-codes**. Communications, IEEE Transactions on, 44(10):1261–1271, Oct 1996.

[COV1991] COVER, T. M.; THOMAS, J.. **Elements of Information Theory**. Wiley, 1991.

[HEA2005] HU, X.; ELEFTHERIOU, E. ; ARNOLD, D.. **Regular and irregular progressive edge-growth tanner graphs**. IEEE Trans. Inform. Theory, 51(1):386–398, Jan. 2005.

[JKM2000] JIN, H.; KHANDEKAR, A. ; MCELIECE, R.. **Irregular repeat-accumulate codes**. In: PROC. OF THE SECOND INTERNATIONAL SYMPOSIUM ON TURBO CODES AND RELATED TOPICS, p. 1–8, Brest, France, September 2000.

[LMS1997] LUBY, M. G.; MITZENMACHER, M.; SHOKROLLAHI, M. A.; SPIELMAN, D. A. ; STEMANN, V.. **Practical loss-resilient codes**. In: STOC '97: PROCEEDINGS OF THE TWENTY-NINTH ANNUAL ACM SYMPOSIUM ON THEORY OF COMPUTING, p. 150–159, New York, NY, USA, 1997. ACM.

[MCT2006] RICHARDSON, T.; URBANKE, R.. **Modern Coding Theory**. Cambridge University Press, October 2006.

[MKY1997] MACKAY, D.; NEAL, R.. **Near shannon limit performance of low density parity check codes**. Electronics Letters, 33(6):457–458, Mar 1997.

[MKY2003] MACKAY, D. J. C.. **Information Theory, Inference, and Learning Algorithms**. Cambridge University Press, 2003.

[RGG1963] GALLAGER, R. G.. **Low-Density Parity-Check Codes**. MIT Press, 1963.

[RSU2001] RICHARDSON, T. J.; SHOKROLLAHI, A. ; URBANKE, R. L.. **Design of capacity-approaching irregular low-density parity-check codes**. IEEE Trans. Inform. Theory, (2):619–637, 2001.

[SBW2003] WICKER, S. B.; KIM, S.. **Codes, Graphs and Iterative Decoding**. Kluwer Academic Publishers, 2003.

[SJR2005] JOHNSON, S.; WELLER, S.. **Constructions for irregular repeat-accumulate codes**. In: INFORMATION THEORY, 2005. ISIT 2005. PROCEEDINGS. INTERNATIONAL SYMPOSIUM ON, p. 179–183, Sept. 2005.

[TJV2003] TIAN, T.; JONES, C.; VILLASENOR, J. ; WESEL, R.. **Construction of irregular ldpc codes with low error floors**. In: PROC. OF THE IEEE INTERNATIONAL CONFERENCE ON COMMUNICATIONS, volumen 5, p. 3125–3129, May 2003.

[TKM2005] MOON, T. K.. **Error Correction Coding – Mathematical Methods and Algorithms**. Wiley Interscience, 2005.

[Tan1981] TANNER, R.. **A recursive approach to low complexity codes**. Information Theory, IEEE Transactions on, 27(5):533–547, 1981.

A
Degree Distribution and Design Rates

This appendix addresses some elements of graph theory that are used elsewhere in this dissertation. Many previously cited references use differing notations, the following sections specify and explain the notation that is used along the text, derive some important equations and highlight one distinction that should be made when considering IRA codes as opposed to classic LDPC codes.

A.1
Some Definitions

In order to avoid ambiguities we will start with some definitions on Graph Theory from [SBW2003].

Definition 11 (Graph) *A Graph is an ordered pair $G = (\mathcal{V}, \mathcal{E})$ of a set of vertices (or nodes) $\mathcal{V}$ and a set of edges $\mathcal{E}$.* ⋄

Definition 12 (Edge) *An edge is defined by a pair of distinct vertices from $\mathcal{V}$. An edge is said to be* incident *on its end vertices, and the two vertices at opposite ends of an edge are said to be* adjacent *or* neighbors *(additionally, we will refer to two neighbor vertices as being* joined *by the common edge). Two or more edges containing one vertex in common are also adjacent.* ⋄

Definition 13 (Path) *A path is a sequence of distinct adjacent edges. Any two vertices that belong to the same path are said to be* connected*. Additionally, a path that starts and ends in the same vertex is a* cycle*.* ⋄

We next define a Bipartite Graph.

Definition 14 (Bipartite Graph) *A bipartite graph $G = (\mathcal{V}_1 \cup \mathcal{V}_2, \mathcal{E})$ is a graph whose vertices can be divided in two disjoint sets such that no two vertices from the same set are adjacent.* ⋄

Richard Michael Tanner proposed using bipartite graphs to combine simple parity checks into more complex codes [Tan1981], and for this reason these graphs are today called *Tanner graphs*. In these graphs, largely used to represent LDPC codes, the vertices are termed *nodes*. A Tanner graph is therefore a graph $H = (\mathcal{V}_v \cup \mathcal{V}_c, \mathcal{E})$ having two disjoint sets of nodes denominated *Variable-Nodes* and *Check-Nodes* respectively. Tanner graphs are often portrayed as two vertical columns of nodes, with the variable-nodes (to the left) represented by circles and the check-nodes (to the right) as the squares.

To illustrate these facts, we will consider a Tanner graph with n variable-nodes (circles) and m check-nodes (squares) such as the graph illustrated in Figure A.1.

Definition 15 (Node Degree) *The number of the edges joining a variable-node to its adjacent check-nodes is the variable-node degree.* $\diamond$

In a Tanner graph, the degree of a node is the number of edges that are incident on this node. The graph in Figure A.1 has nodes with varying degrees on the left, while all nodes to the right are degree-4 nodes. Such graphs, with variable degree nodes are referred to as an *irregular graph*. Specifically, the graph on Figure A.1 is a left-irregular and right-regular graph. *Regular graphs*, on the other hand, are graphs that are regular on both sides.

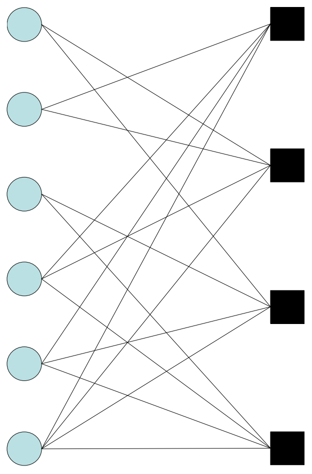

Figure A.1: A small (rate $\frac{1}{3}$) Tanner graph with $\mathbf{\Lambda} = \{\Lambda_1 = 0, \Lambda_2 = 0.5, \Lambda_3 = 1/3, \Lambda_4 = 1/6\}$ and $\mathbf{R} = \{0, 0, 0, 1\}$

Modern codes are randomly generated codes that can be associated to a graph selected from an ensemble of codes with specified statistics. In the characterization of a graph, a parameter that plays an important role is the *degree vector*: a vector with integer components no greater than ℓ_{max} (belonging to the set of integers $\mathbb{Z}_{\ell_{max}} = \{1, \ldots, \ell_{max}\}$), represented by $\mathbf{D}_{\text{left}} = \{D_{\text{left}1}, \ldots, D_{\text{left}n}\}$, where $D_{\text{left}i}, i \in \mathbb{Z}_n$ is the degree of the i-th left-node (equivalently, $\mathbf{D_{right}}$ and $D_{\text{right}i}$ for the right-nodes). Observing the statistical distribution of the values in $\mathbf{D}$ we establish the graph's *degree distribution*.

The graph's degree distribution is the statistical property that provides the upper bounds for the error correcting capability of codes under the message passing algorithm in a given ensemble. The degree distribution can be defined either from the perspective of the nodes or from the perspective of the edges. First we define it from the perspective of nodes: The degree distribution vectors $\mathbf{\Lambda} = \{\Lambda_1, \Lambda_2, \ldots, \Lambda_{D_l max}\}$ and $\mathbf{R} = \{R_1, R_2, \ldots, R_{d_r max}\}$, where Λ_i or R_i are, respectively, the fractions of variable and check nodes with degree i. Therefore, $\mathbf{\Lambda}$ and $\mathbf{R}$ are the discrete probability masses.

$$\Lambda_i = P[d_{lx} = i] \tag{A-1}$$

$$R_i = P[d_{rx} = i] \tag{A-2}$$

Despite the simplicity of the definition above, the studies of Density Evolution and Belief Propagation require an observation from the perspective of the graph's edges instead of the nodes. Therefore, we introduce the notation of edge-degree distributions. Every edge has a left-degree and a right-degree: The left-degree is the degree of the variable-node in one end of the edge (the left-end), and the right-degree is the degree of the check-node at the opposite end. Analogously to the degree vector $\mathbf{D}$ described above, we can define an edge-degree vector $\mathbf{d}_{\text{left}} = \{d_{\text{left}1}, \ldots, d_{\text{left}|\mathcal{E}|}\}$ with the left-degrees of every edge in the graph. So we define the degree distribution from the perspective of edges

$$\lambda_i = P[d_{\text{left}x} = i], \tag{A-3}$$

which is the fraction of edges in $\mathcal{E}$ with left degree i. Alternatively, for the right degree of an edge

$$\rho_i = P[d_{\text{right}x} = i]. \tag{A-4}$$

Knowing the degree distribution of a graph from the perspective of the nodes, it becomes easy to obtain the degree distribution from the perspective of edges simply by verifying that the number of edges with left-degree i is given by $ni\Lambda_i$. Hence,

$$\lambda_i = \frac{i\Lambda_i}{\sum_j j\Lambda_j} \tag{A-5}$$

$$\rho_i = \frac{iR_i}{\sum_j jR_j}, \tag{A-6}$$

meaning the fraction of edges connected to a degree-i variable-node (A-5) or check-node (A-6).

Using the notions that $\lambda_i \propto i\Lambda_i$ and that degree distributions are probability densities (therefore, should sum to 1) we can alternatively define Λ_i in terms of λ_i.

$$\Lambda_i = \frac{\lambda_i/i}{\sum_j \lambda_j/j} \tag{A-7}$$

$$R_i = \frac{\rho_i/i}{\sum_j \rho_j/j}. \tag{A-8}$$

Substituting the expression in (A-5) for λ_i in (A-7), it becomes clear that $\sum_j j\Lambda_j = \left(\sum_j \lambda_j/j\right)^{-1}$, which is, by definition, the average value of the variable node degree a_l.

$$a_l \triangleq \sum_j j\Lambda_j \tag{A-9}$$

$$a_r \triangleq \sum_j jR_j \tag{A-10}$$

We now define the degree polynomials, often used to express extrinsic information functions:

$$\lambda(x) = \sum_i \lambda_i x^{i-1}$$

$$\rho(x) = \sum_i \rho_i x^{i-1},$$

whose integral in the interval $[0, 1]$ can replace the denominator in (A-7) and (A-8). This provides us with a more elegant expression for the average node degrees a_l — for variable nodes — and a_r —parity check nodes — taken from the values of the edge degree distributions.

$$a_l = \left(\int_0^1 \lambda(t)dt\right)^{-1} \tag{A-11}$$

$$a_r = \left(\int_0^1 \rho(t)dt\right)^{-1} \tag{A-12}$$

A.2 Design Rates – IRA and LDPC

One subtle difference between IRA and LDPC codes should be noted when studying those codes. While LDPC codes are based on bipartite graphs that oppose variable nodes to check nodes (see figure A.1), the graphs that represent IRA codes, although still bipartite, demand that a clear distinction is made between parity nodes — i.e. the variable nodes that contain the redundant information in a systematic code — and the information-nodes that carry the original message. For this reason, IRA codes are graphically depicted as three columns of nodes (see figure A.2), with the parity nodes (circles) to the right of the check nodes.

In a classic LDPC graph, the nodes depicted at the left-hand column comprise the totality of variable nodes, while in an IRA code we define the left-hand column as the information nodes only. All parity nodes excepted the last one have degree two, and in this work — following the convention in [JKM2000] — they are not considered in the degree distributions, since their edges are traced following a specific pattern while all others are randomly permuted. In other words, a degree distribution where $\lambda_2 = 0$ has no information nodes with degree two, but it does have $m-1$ parity nodes with degree two representing the redundant bits. Other works, such as [SJR2005] do not follow this convention.

This can be easily observed when comparing the graphs in figures A.1 and A.2. Their degree distributions are in fact different and so are the block-lengths and code-rates, but the degree-distribution vector $\mathbf{\Lambda}$ is the same because A.2 is an IRA code and follows a different notation.

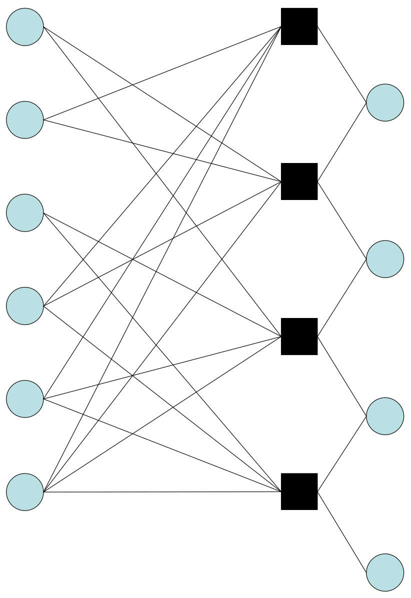

Figure A.2: A small (rate $\frac{3}{5}$) graph of an IRA code with $\Lambda = \{0, 0.5, 1/3, 1/6\}$ and $\mathbf{R} = \{0, 0, 0, 1\}$

A.2.1 LDPC

We define the total number of edges in the graph and the design rate of a code

$$|E| = ma_r = na_l, \quad r = \frac{k}{n} = \frac{n-m}{n}.$$

Now we can express the rate as a function of the graph's degree distributions, independent of the chosen block-length.

$$\begin{aligned} r &= 1 - \frac{m}{n} \\ &= 1 - \frac{E/a_r}{E/a_l} \\ &= 1 - \frac{\int_0^1 \rho(t)dt}{\int_0^1 \lambda(t)dt} \end{aligned} \tag{A-13}$$

A.2.2 IRA

If we consider only the edges that join check-nodes to information-nodes, we have

$$E = ma_r = ka_l, \quad r = \frac{k}{k+m}$$

and a different expression for the design rate as a function of the graph's degree distributions:

$$\begin{aligned} r &= \left(1 + \frac{m}{k}\right)^{-1} \\ &= \left(1 + \frac{E/a_r}{E/a_l}\right)^{-1} \\ &= \left(1 + \frac{\int_0^1 \rho(t)dt}{\int_0^1 \lambda(t)dt}\right)^{-1} \end{aligned} \tag{A-14}$$

There is no theoretical reason, however, for the distinction between parity nodes and check-nodes. In [SJR2005], IRA codes are just defined as a special case of LDPC codes where $k\Lambda_2 \geq m$ to ensure there will be at least m parity nodes with degree two.

B
Message Passing Algorithm

Algorithm 6 is a concise description of the message passing algorithm using the notation defined in chapter 3.

Algorithm 6 Message Passing Algorithm

Inputs : $\boldsymbol{\mu}^0(0)$ and $\boldsymbol{\mu}^0(1)$ such that $\mu_j^0(x) = P\left(C_j = x \mid y_j\right), \forall j \in [1, n]$ L is the upper limit on the number of iterations

$$\mu_j^0(1) = (1 + e^{-2ay_j/\sigma^2})^{-1}; \quad \mu_j^0(0) = (1 + e^{2ay_j/\sigma^2})^{-1}$$

```
Ensure: $\mathbf{H} \cdot \hat{\mathbf{x}}^{\mathrm{T}} = \mathbf{0}$
  $l = 0$;
  $\forall\{i \in [1, m], j \in [1, n]\}$, $\mu_{i,j}^0 = \mu_j^0$;
  repeat
    $l = l + 1$;
    Horizontal Step:
    for $i = 1$ to $m$ do
      $\delta\gamma_i = \prod_{j' \in \mathcal{N}_i} \delta\mu_{i,j'}^{l-1}$;
      {Check-nodes receive messages from the variable-nodes}
      for every $j \in \mathcal{N}_i$ do
        $\delta\gamma_{i,j} = (\delta\gamma_i)/(\delta\mu_{i,j}^{l-1})$;
        $\gamma_{i,j}(0) = (\delta\gamma_{i,j} + 1)/2$;
        $\gamma_{i,j}(1) = (1 - \delta\gamma_{i,j})/2$;
        {The check-nodes compute the messages and send them back to the
        variable-nodes}
      end for
    end for
    Vertical Step:
    for $j = 1$ to $n$ do
      $\alpha_j(0) = \mu_j^0(0) \cdot \prod_{i' \in \mathcal{M}_j} \gamma_{i',j}(0)$;
      $\alpha_j(1) = \mu_j^0(1) \cdot \prod_{i' \in \mathcal{M}_j} \gamma_{i',j}(1)$;
      {Variable-nodes receive messages from the check-nodes}
      for every $i \in \mathcal{M}_j$ do
        $\alpha_{i,j}(0) = \alpha_j(0)/\gamma_{i,j}(0)$;
        $\alpha_{i,j}(1) = \alpha_j(1)/\gamma_{i,j}(1)$;
        $\mu_{i,j}^l(0) = \alpha_{i,j}(0)/[\alpha_{i,j}(0) + \alpha_{i,j}(1)]$;
        $\mu_{i,j}^l(1) = \alpha_{i,j}(1)/[\alpha_{i,j}(0) + \alpha_{i,j}(1)]$;
        $\delta\mu_{i,j}^l = \mu_{i,j}^l(0) - \mu_{i,j}^l(1)$;
        { The variable-nodes compute the messages and send them back to
        the check-nodes}
      end for
      {Compute pseudo-posterior probabilities}
      $\mu_j^l(0) = \alpha_j(0)/[\alpha_j(0) + \alpha_j(1)]$;
      $\mu_j^l(1) = \alpha_j(1)/[\alpha_j(0) + \alpha_j(1)]$;
      {These values will be used for hard decision and are not used for
      computing messages}
      $\hat{\mathbf{x}} = \lceil \boldsymbol{\mu}^l(1) \rceil$;
    end for
  until $\mathbf{H} \cdot \hat{\mathbf{x}}^{\mathrm{T}} = \mathbf{0}$ or $l > L$
```

C Channel Capacity

The aim of channel coding is to provide reliable communication at rates close to the channel capacity, as established by Shannon in 1948. In this work, we focus on binary codes applied to Binary Phase-Shift Keying (BPSK) modulation, therefore we want to achieve the capacity of a binary constrained channel. In the following sections we will expose some of the theory behind the channel capacity for the binary constrained and the unconstrained channel.

C.1 The Shannon Limit

We define the channel capacity as the maximum of the information rates that can be transmitted through a channel.

$$C = \max_{p(x)} I(X;Y) = \max_{p(x)} H(Y) - H(Y \mid X), \tag{C-1}$$

given a discrete memoryless channel, i.e. a system whose output Y is related to the input X by a known conditional probability density $p(Y \mid X)$.

In a Binary Symmetric Channel with error probability p_e, the channel capacity can easily be computed by applying (C-1) directly, giving

$$C = \max_{p(x)} H(Y) - H(p_e) = 1 - H(p_e). \tag{C-2}$$

Although IRA codes operate on the AWGN channel with soft-decision decoding, this result helps us establish the error floors for transmission at rates below channel capacity.

C.2 The Gaussian Channel

The discrete time AWGN channel, as mentioned in chapter 1, has a real valued input X and a real valued output $Y = X + Z$, where $Z \sim \mathcal{N}(0, \frac{\mathcal{N}_0}{2})$ and $\frac{\mathcal{N}_0}{2}$ (the bilateral noise spectral density) is a known variance. It is also implied that the input has a power constraint which limits the communication rates.

Using equation (C-1) in this context we can see that $H(Y \mid X) = H(Z)$, since the noise is independent from the source. Also, since it operates on real valued variables, the capacity of the AWGN channel is a function of the differential entropy of the input and output variables, represented by $h(\cdot)$. So we write instead

$$C = \max_{p(x):E[X^2]\leq P} h(Y) - h(Y \mid X) \tag{C-3}$$
$$= h(Y) - h(Z), \tag{C-4}$$

where $h(Z) = \frac{1}{2}\log(2\pi e\frac{\mathcal{N}_0}{2})$, the differential entropy of the normal distribution.

It can be proven ([COV1991], chapter 9) that the normal distribution is the one that maximizes the entropy for a given variance, and the sum of two gaussian random variables is a gaussian random variable. Thus, we maximize the mutual information of the AWGN channel by making $Y \sim \mathcal{N}(0, \frac{\mathcal{N}_0}{2} + P)$. This gives us the capacity of the unconstrained AWGN channel.

$$\begin{aligned} C &= \tfrac{1}{2}\log\left(2\pi e(\tfrac{\mathcal{N}_0}{2} + P)\right) - \tfrac{1}{2}\log\left(2\pi e\tfrac{\mathcal{N}_0}{2}\right) \\ &= \tfrac{1}{2}\log(1 + \mathit{SNR}) \text{ bits per channel use,} \end{aligned} \tag{C-5}$$

where $\mathit{SNR} = 2R\frac{E_b}{\mathcal{N}_0}$ and R stands for the total transmitted bit rate.

The simulations presented in this work, on the other hand, use a BPSK modulation scheme which is sure to never achieve this capacity. This is due to the fact that the probability distribution of the detected value $p(Y) = \frac{1}{2}(p(Y \mid X = 1) + p(Y \mid X = -1))$ will not assume the shape of a normal distribution. The capacity of the binary constrained channel is computed numerically using the Blahut-Arimoto algorithm [COV1991] with a discretized approximation of the channel posteriori probabilities $p(Y \mid X = 1)$ and $p(Y \mid X = -1)$ as inputs.

C.2.1 The Channel Coding Theorem

The knowledge of the channel capacity allows us to set lower bounds on the bit-error probabilities of communication systems using channel coding. Using Fano's Inequality or Rate-Distortion Theory, ([BEB1998], chapter 3, section 3.3; [MKY1997], chapter 10, section 10.5) we can establish the code rate necessary to communicate at a given bit-error rate above channel capacity

$$\mathit{Rate} = \frac{C}{1 - H_b(p_{be})}, \tag{C-6}$$

where p_{be} is the minimum bit-error rate after decoding.

Considering a fixed rate, (C-6) combined with (C-5) gives us Figure C.1. Nevertheless, the simulations performed for this work did not use the unconstrained AWGN channel. As explained in Chapter 1, all simulations were modeled on the binary channel, so the bounds used in the individual performance curves (as in Appendix D) were obtained from the Blahut-Arimoto algorithm [COV1991], as seen in Figure C.2.

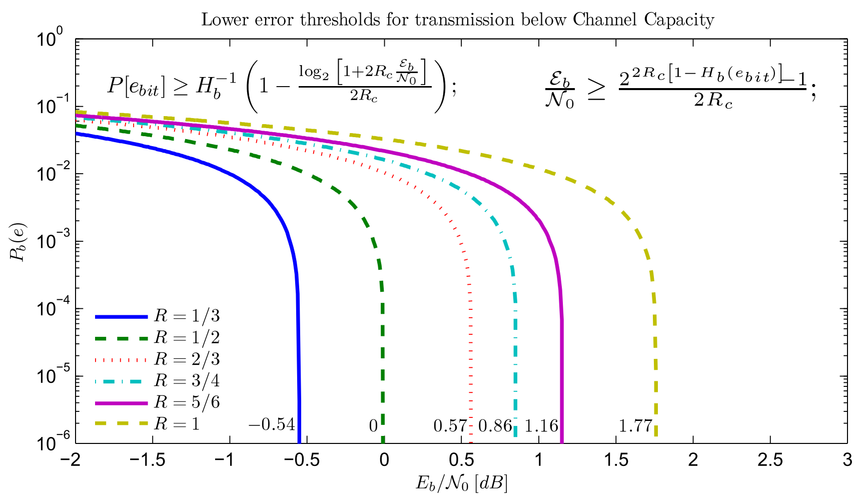

Figure C.1: Lower bit-error rate bounds for the unconstrained AWGN channel channel

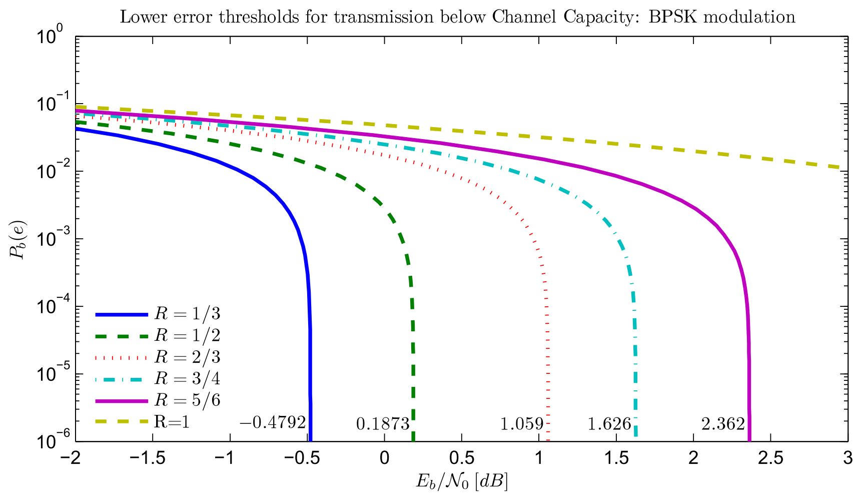

Figure C.2: Lower bit-error rate bounds for the binary constrained AWGN channel channel

D Complete Plots

Here we present a wider selection of performance plots from our simulations. The legend in each plot identifies the various curves that aid us in understanding the code's performance:

Bit errs: The thick solid blue lines with star-shaped marks show the total bit-error probabilities that were estimated from simulation.

Undet. (bit): The red dotted lines with triangular marks pointing downwards represent the estimated undetected bit-error probabilities. These lines may be absent in plots showing the performance of codes where no undetected errors occurred. More on undetected errors is explained in Chapter 4.

Det. (bit) The purple dotted lines with triangular marks pointing upwards discriminate the estimated **detected** bit-error probabilities. In most cases (when most errors are detected by a decoder failure declaration) this line will be superposed to the blue line, unless a high rate of undetected errors occur.

Sh. Lim. The solid thin black line shows the Shannon limit, i.e. the theoretical lower bound for bit-errors for the given code rate in a binary constrained channel (see Figure C.2 in Appendix C).

Raw BPSK The red dash-dotted line shows the performance of an uncoded transmission on the same channel. Since the x-axis give the channel conditions in terms of $\frac{E_b}{\mathcal{N}_0}$, a bad code can be outperformed by uncoded transmission, where no energy is wasted on redundant bits.

max Pbe (0.05) The cross-marks were to show the reader how reliable are the points in the total bit-error performance curves. There is a probability of 0.05 that the actual bit error probability is above that mark. Those marks are more distanced from the blue solid line at points where less than ten error events could be obtained from our simulations. Our simulations were timed out after more than twelve hours with an insufficient number of error-events.

The value of the confidence interval δ, such that $P_{\text{be}}^{\max} = P_{\text{be}} + \delta$, is obtained as following

$$\delta = Q^{-1}(0.05) \times \sqrt{\frac{P_{\text{be}} \times (P_{\text{be}} - 1)}{N \times k}}, \tag{D-1}$$

where $Q^{-1}(\cdot)$ is the inverse function to $Q(x) = \int_x^\infty \exp(\frac{u^2}{2})$, the complement to the cumulative distribution function of the zero-centered unitary variance Normal distribution. The expression under $\sqrt{\cdot}$ is the estimate to the variance of the samples (the error events) considering $N \times k$ trials: N transmitted blocks containing k-bits each.

D.1 Standard PEG

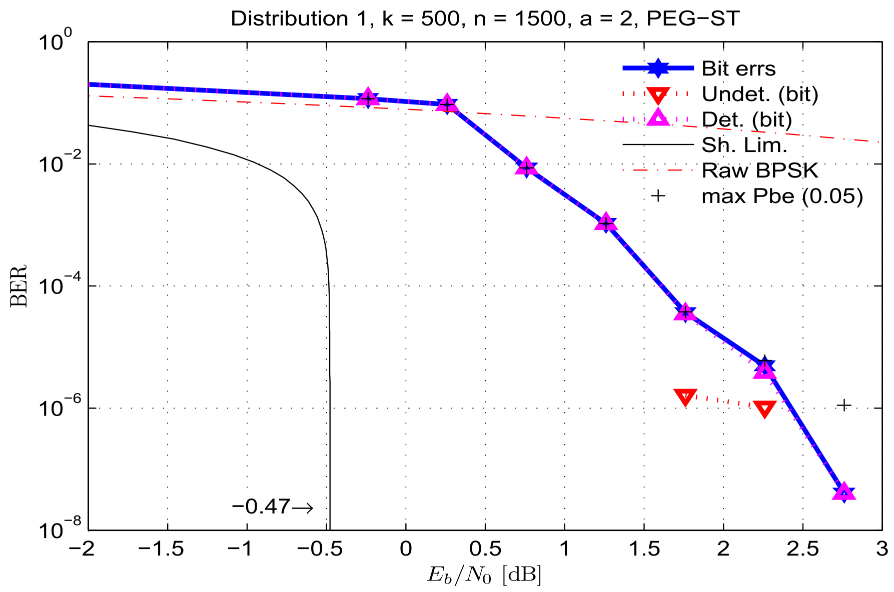

Figure D.1: Code performance using degree distribution #1 with the standard PEG algorithm

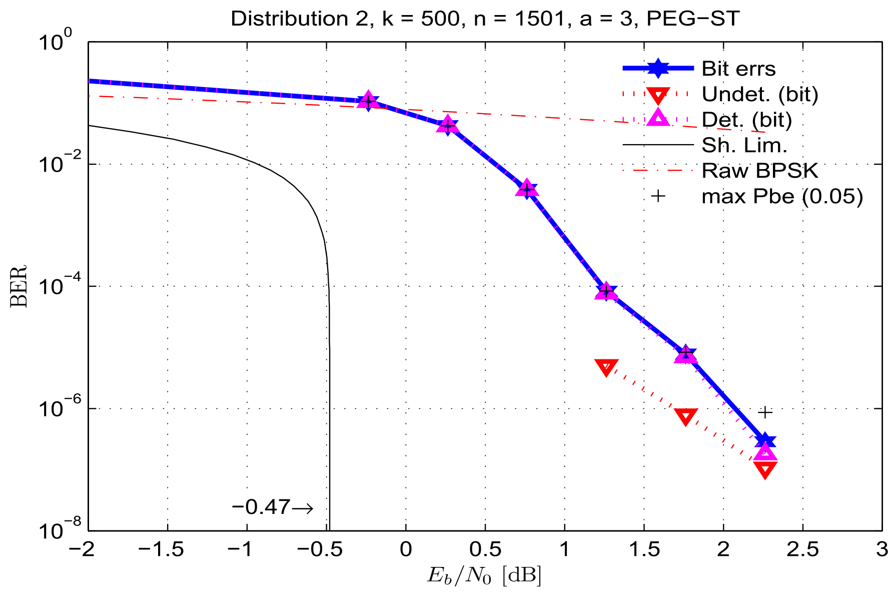

Figure D.2: Code performance using degree distribution #2 with the standard PEG algorithm

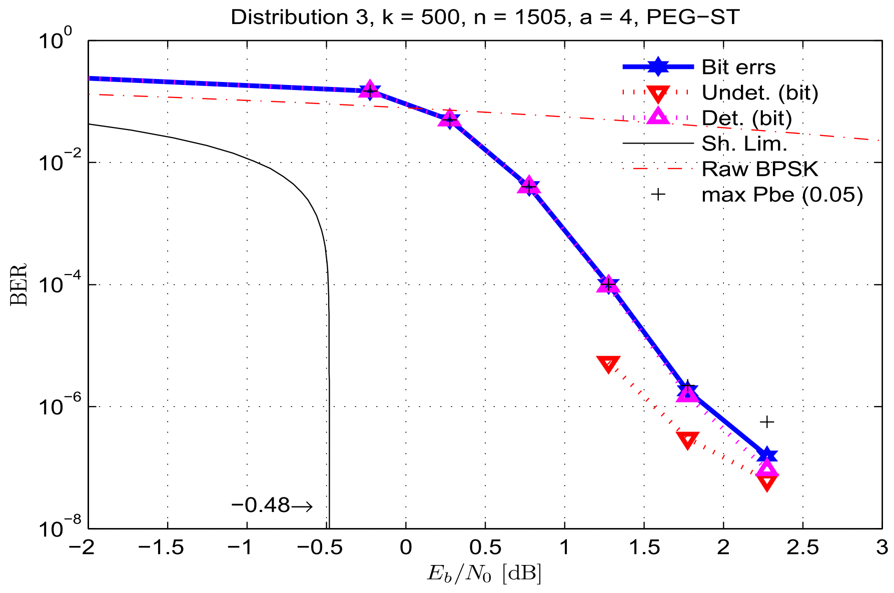

Figure D.3: Code performance using degree distribution #3 with the standard PEG algorithm

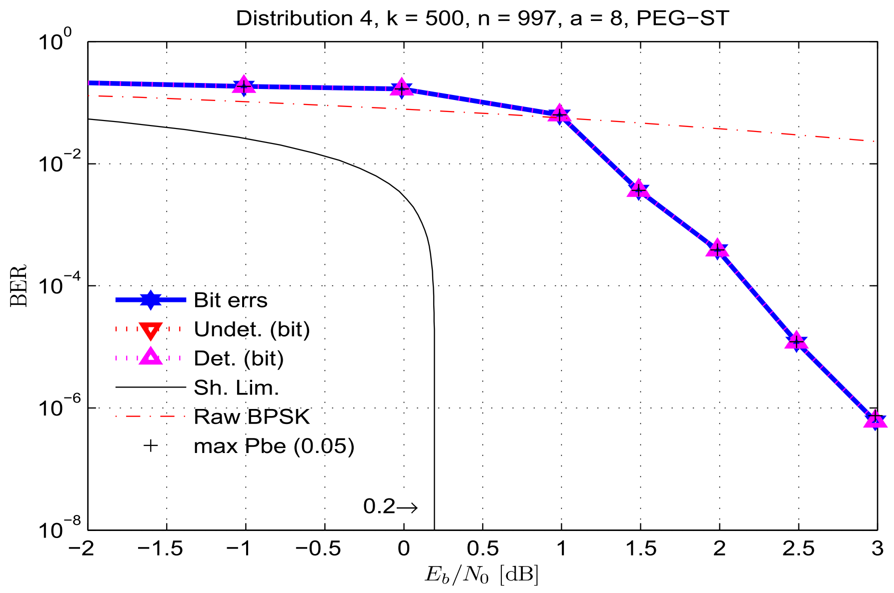

Figure D.4: Code performance using degree distribution #4 with the standard PEG algorithm

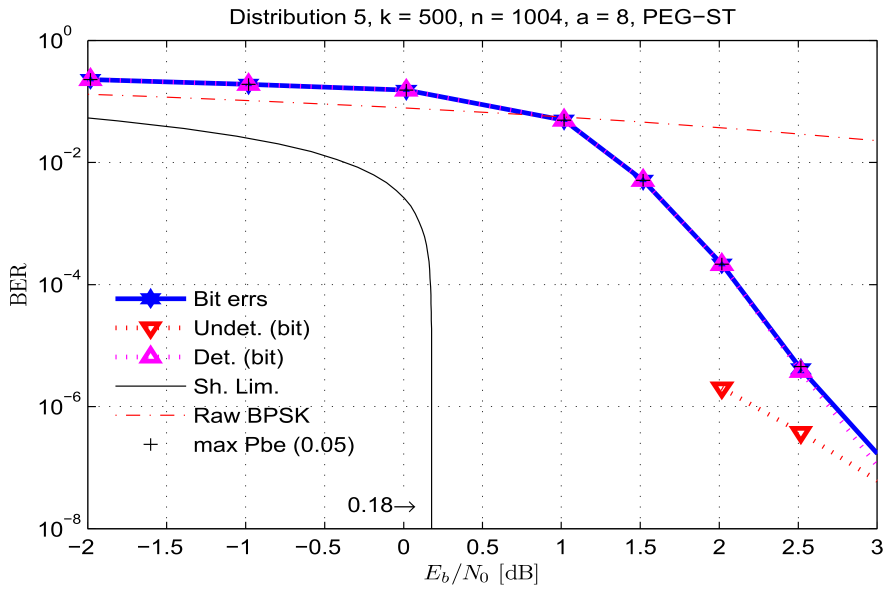

Figure D.5: Code performance using degree distribution #5 with the standard PEG algorithm

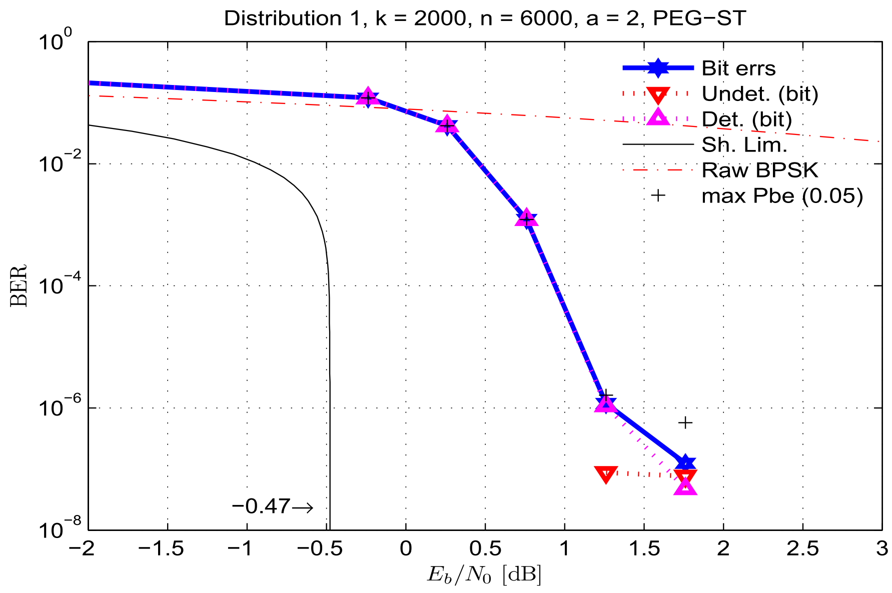

Figure D.6: Code performance using degree distribution #1 with the standard PEG algorithm

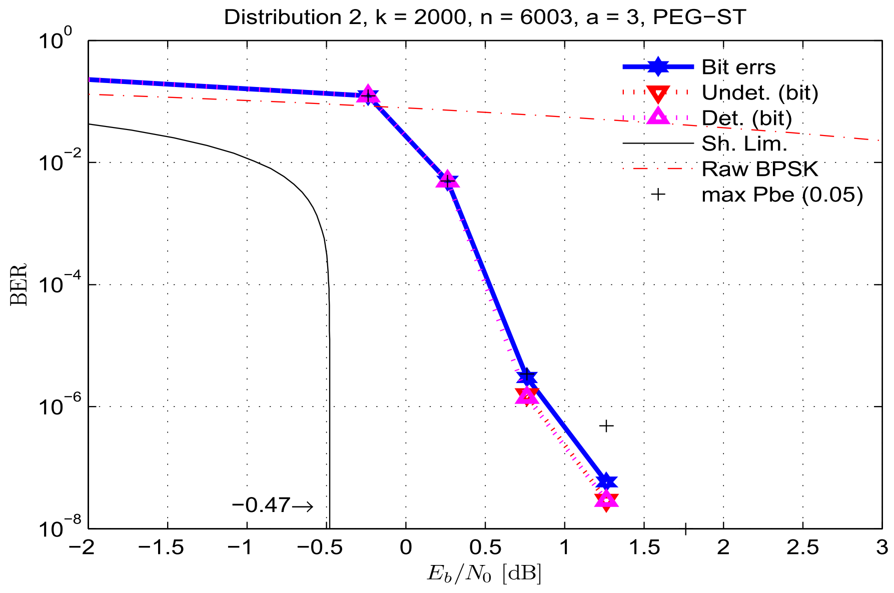

Figure D.7: Code performance using degree distribution #2 with the standard PEG algorithm

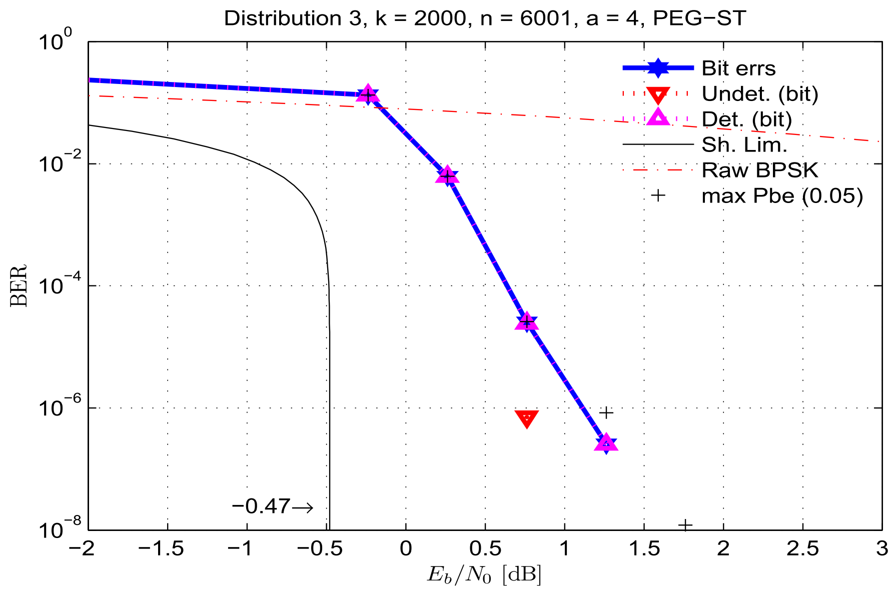

Figure D.8: Code performance using degree distribution #3 with the standard PEG algorithm

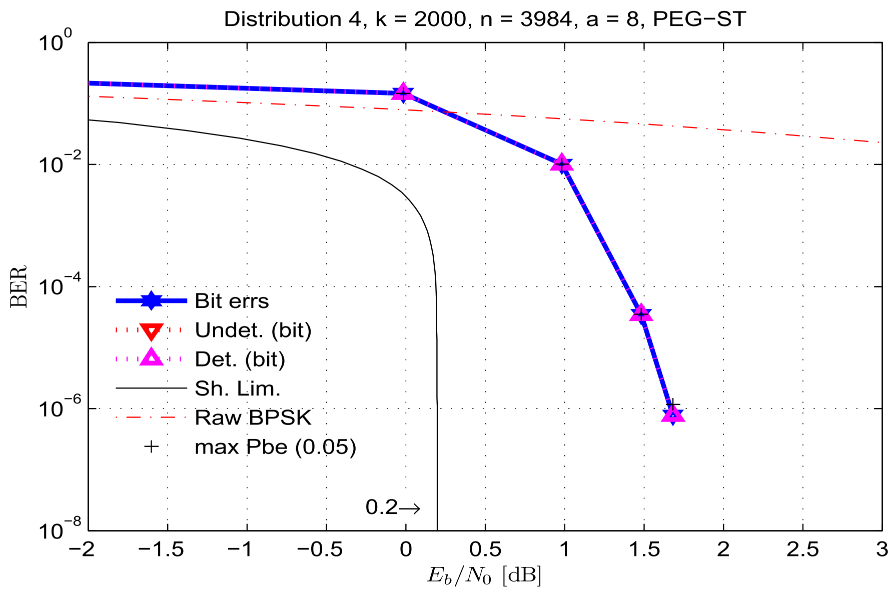

Figure D.9: Code performance using degree distribution #4 with the standard PEG algorithm

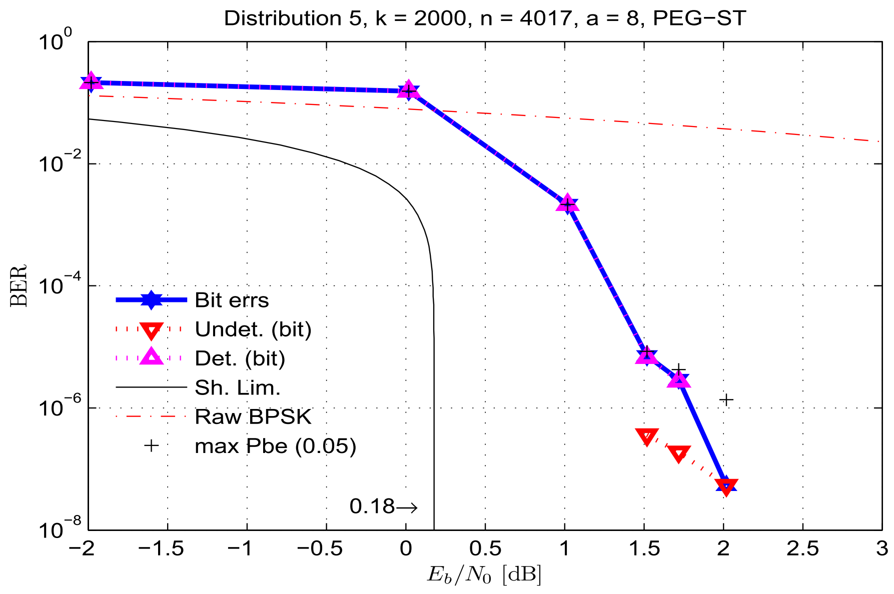

Figure D.10: Code performance using degree distribution #5 with the standard PEG algorithm

D.2
Look-Ahead Enhanced PEG

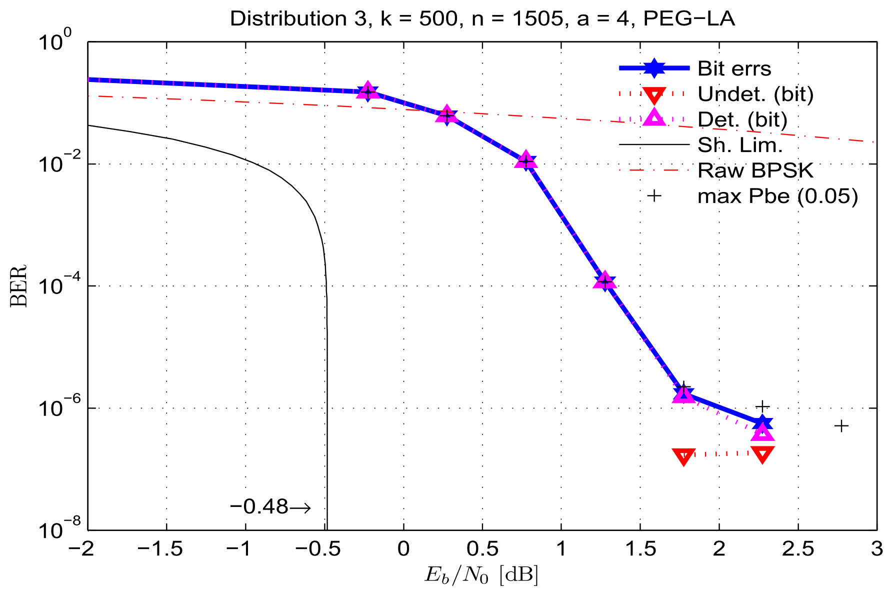

Figure D.11: Code performance using degree distribution #3 with the PEG-LA algorithm

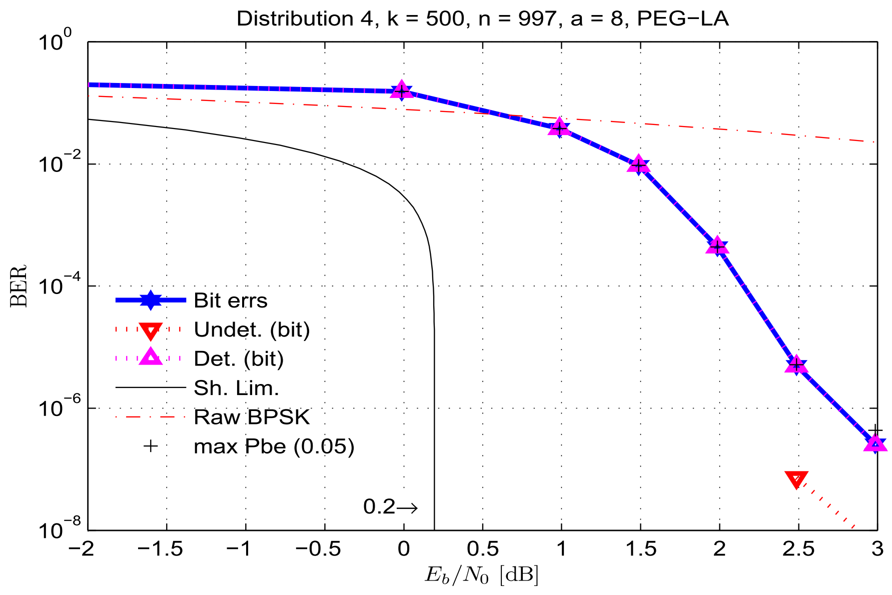

Figure D.12: Code performance using degree distribution #4 with the PEG-LA algorithm

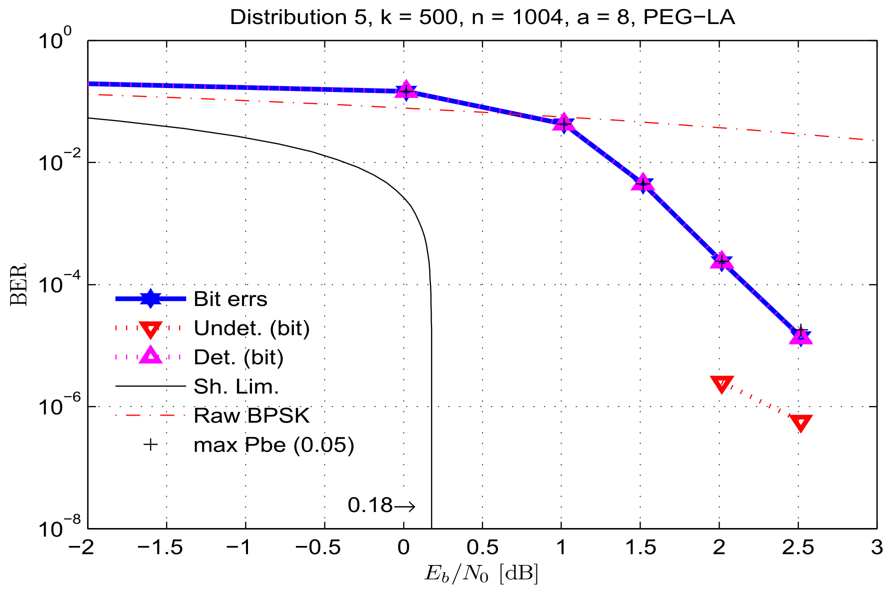

Figure D.13: Code performance using degree distribution #5 with the PEG-LA algorithm

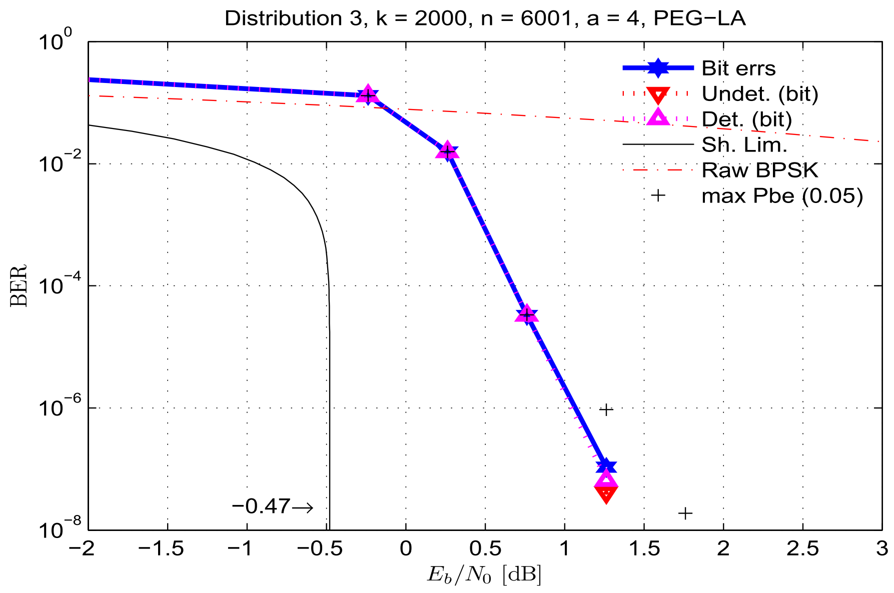

Figure D.14: Code performance using degree distribution #3 with the PEG-LA algorithm

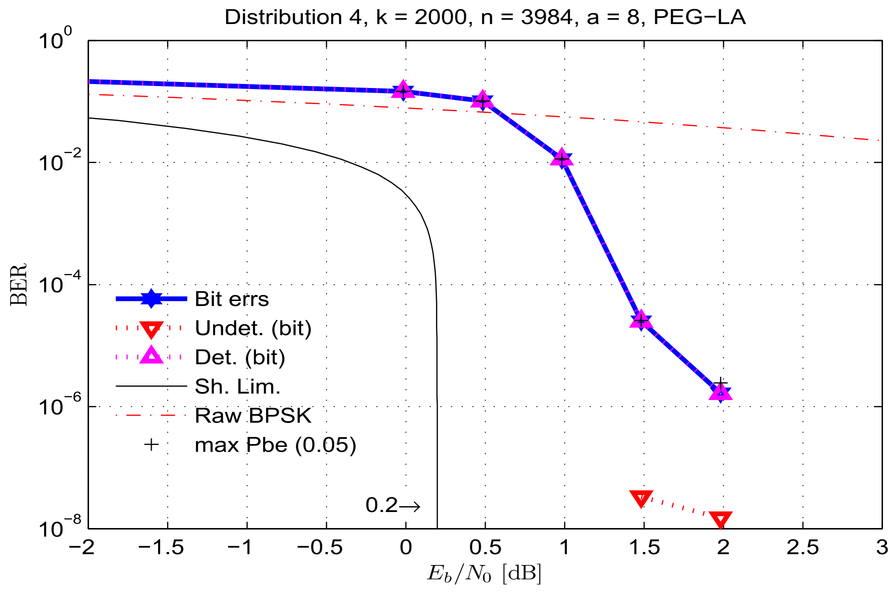

Figure D.15: Code performance using degree distribution #4 with the PEG-LA algorithm

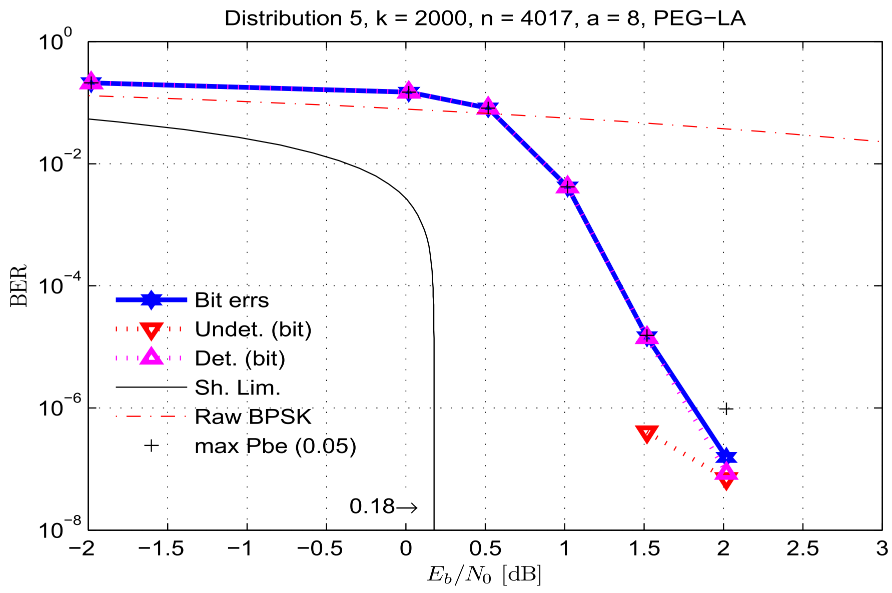

Figure D.16: Code performance using degree distribution #5 with the PEG-LA algorithm

D.3 Reverse Look-Ahead PEG

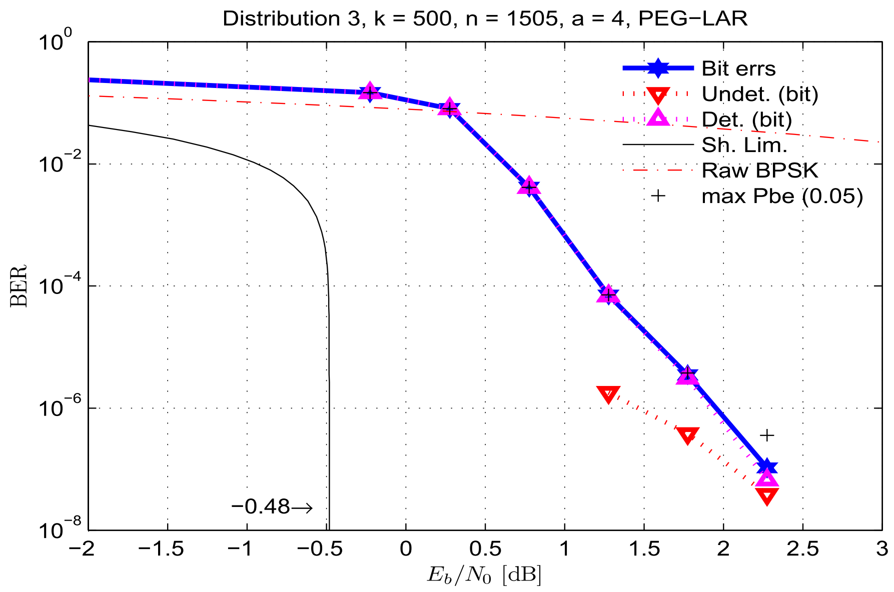

Figure D.17: Code performance using degree distribution #3 with the PEG-LAR algorithm

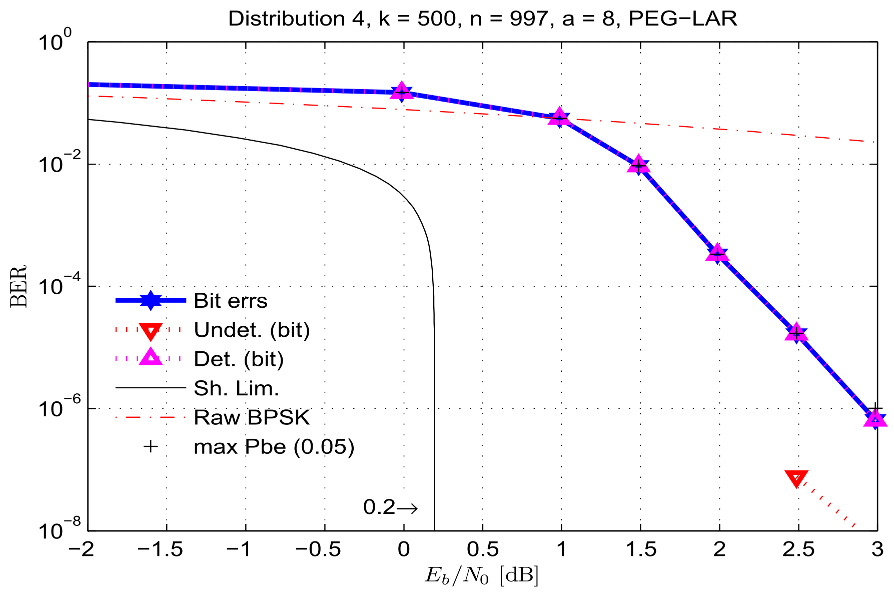

Figure D.18: Code performance using degree distribution #4 with the PEG-LAR algorithm

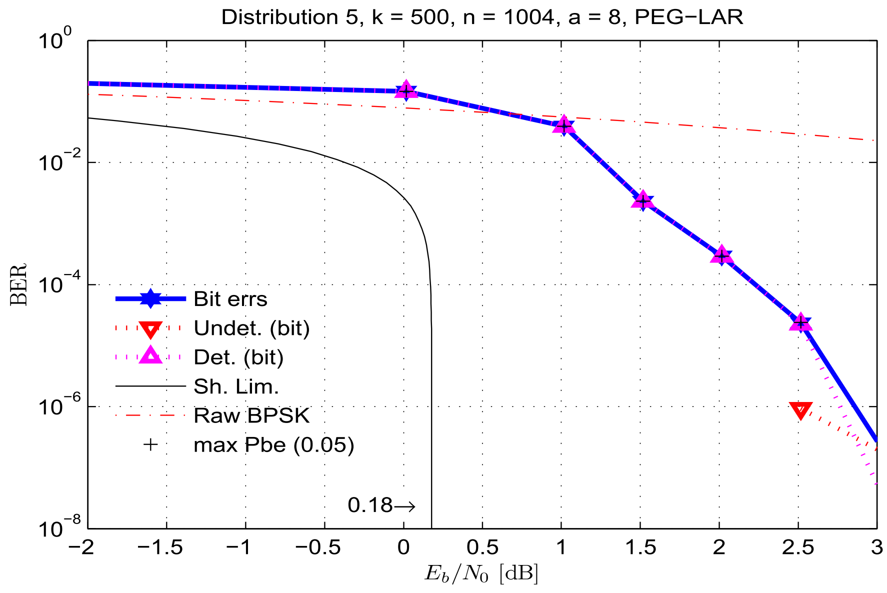

Figure D.19: Code performance using degree distribution #5 with the PEG-LAR algorithm

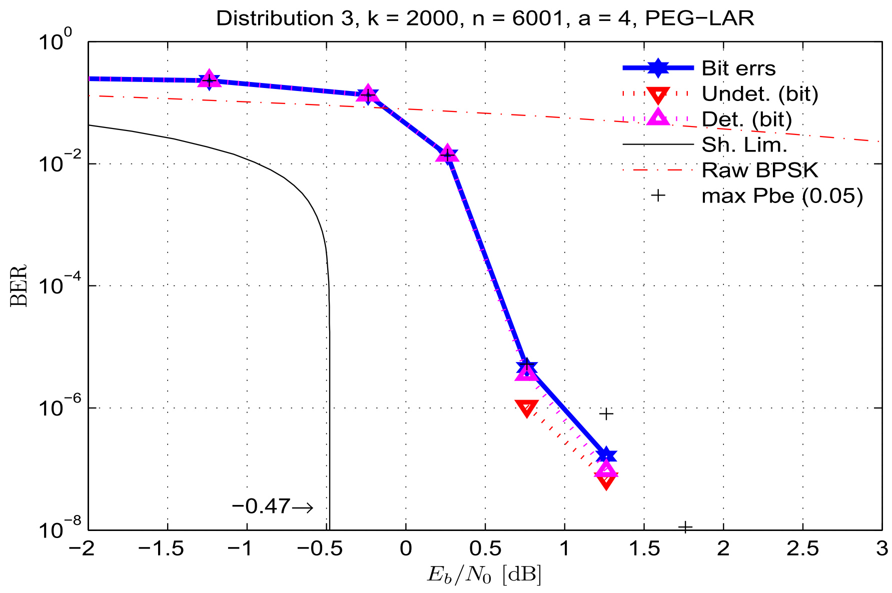

Figure D.20: Code performance using degree distribution #3 with the PEG-LAR algorithm

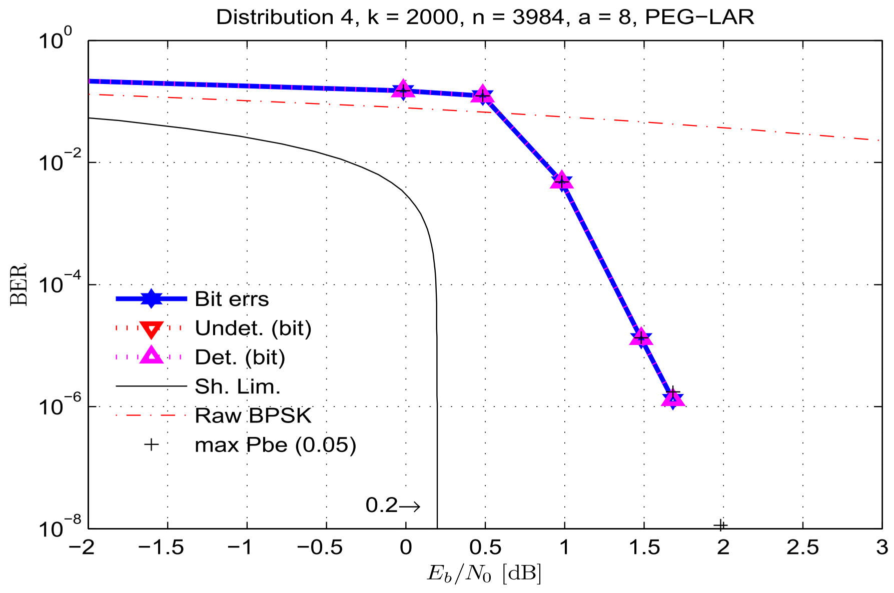

Figure D.21: Code performance using degree distribution #4 with the PEG-LAR algorithm

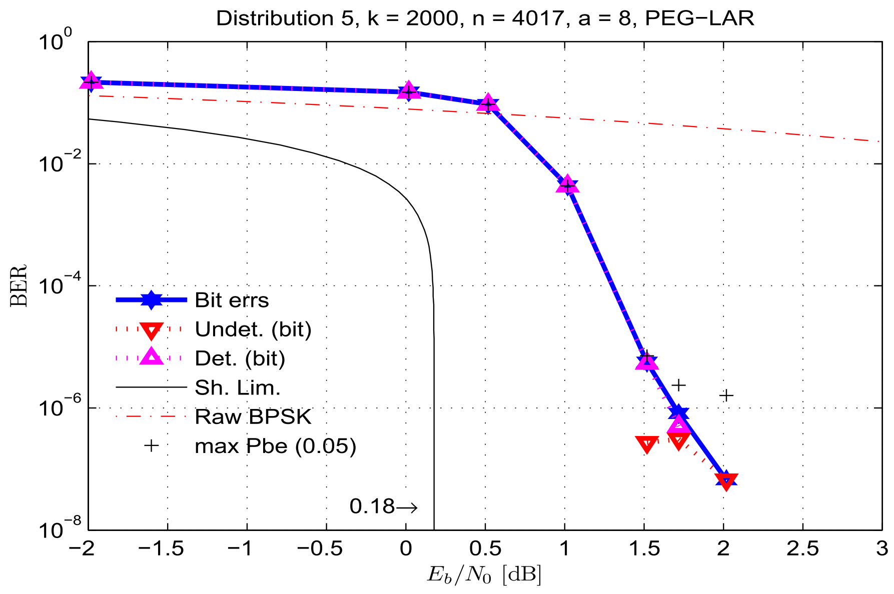

Figure D.22: Code performance using degree distribution #5 with the PEG-LAR algorithm