LDPC CODES FOR IRIG-106 WAVEFORMS: PART I–CODE DESIGN

Erik Perrins
Department of Electrical Engineering and Computer Science
University of Kansas
esp@ieee.org

ABSTRACT

Low density parity check (LDPC) codes allow a communications link to operate reliably at signal to noise ratios that are very close to the Shannon limit. Because of this, in the early 2000s they were studied in connection with SOQPSK-TG and were eventually adopted into the IRIG-106. The deployment for SOQPSK-TG has proved to be very successful, which has motivated interest in finding an LDPC solution for PCM/FM and ARTM CPM. Such a solution, however, has proved to be elusive for reasons that were not entirely clear in the past. In this paper, we lay out the fundamental considerations that must be made in order to design LDPC codes for a specific modulation format. In doing so, we show that SOQPSK-TG enjoys specific similarities with BPSK that allowed an "easy path" toward an LDPC solution in IRIG-106. Most importantly, we show that when the design process begins at the proper starting point, it is just as easy to design LDPC codes that are customized to a particular modulation. We then apply this straightforward design process to PCM/FM and ARTM CPM and demonstrate that **the resulting LDPC codes perform around one dB from the respective channel capacities of these modulations**. In our companion paper, we develop parallel decoder architectures for these schemes that can achieve high throughput. As such, these codes can be considered to fill in the options for LDPC codes that are currently absent in the IRIG-106 standard.

INTRODUCTION

The desired LDPC coding scheme is shown in Figure 1 and consists of an LDPC encoder and a CPM modulator that are separated by an interleaver (denoted by the symbol Π). In our companion paper [1], we develop parallel decoder architectures for this system that can achieve high throughput. For the continuous phase modulations (CPMs), we consider pulse code modulation/frequency modulation (PCM/FM), the Telemetry Group version of shaped-offset quadrature phase shift keying (SOQPSK-TG), and the multi-h CPM developed by the Advanced Range TeleMetry program (ARTM CPM), which are all defined in full detail in the IRIG-106 standard [2]. For convenience, going forward we refer to these, respectively, as ARTM0, ARTM1, and ARTM2.

Our approach to designing LDPC codes for the CPM-based scheme in Figure 1 draws primarily from the references [3–6] with modification and adaptation as needed. A critical step is that the design philosophy must begin with a characterization of the phenomenon known as *extrinsic information transfer (EXIT)* [7] that pertains to the CPM scheme itself. The EXIT properties of the CPM waveform are thus central to the formation of the matching LDPC code. We focus only on *quasi-cyclic* LDPC codes, primarily because the existing IRIG-106 codes are of this type, but also because they enjoy the twin advantages of good performance and low encoding complexity. We

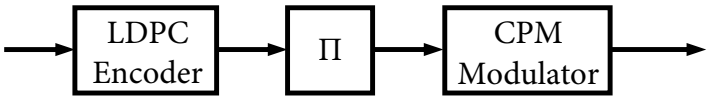

Figure 1: LDPC-coded CPM Model.

use the protomatrix EXIT (PEXIT) design technique [3] because of its simplicity *and* accuracy in predicting the final performance of the code. Furthermore, a number of additional design considerations (such as error floor performance) can be incorporated into the PEXIT design process.

PROTOMATRIX BASICS

A very large quasi-cyclic LDPC code can be specified by a relatively small *protomatrix* (or *protograph*), which is then "lifted" to form the much larger, final version of the parity check matrix for the code. The protomatrix has the following form

$$\mathbf{B} = \begin{pmatrix} b_{0,0} & b_{0,1} & \cdots & b_{0,N_{\mathrm{B}}-1} \\ b_{1,0} & b_{1,1} & \cdots & b_{1,N_{\mathrm{B}}-1} \\ \vdots & \vdots & \ddots & \vdots \\ b_{M_{\mathrm{B}}-1,0} & b_{M_{\mathrm{B}}-1,1} & \cdots & b_{M_{\mathrm{B}}-1,N_{\mathrm{B}}-1} \end{pmatrix} \tag{1}$$

which has dimensions $M_{\mathrm{B}} \times N_{\mathrm{B}}$. The columns of $\mathbf{B}$ correspond to "variable nodes" and we index the columns with $0 \leq j \leq N_{\mathrm{B}} - 1$. The rows of $\mathbf{B}$ correspond to "check nodes" and we index the rows with $0 \leq i \leq M_{\mathrm{B}} - 1$. The elements of $\mathbf{B}$ are drawn from a $\mathcal{B}$-ary alphabet, i.e. $b_{i,j} \in \{0, 1, \ldots, \mathcal{B} - 1\}$. The sum, or "weight," of the j-th column is referred to as the variable node degree, and likewise for the degree of the i-th check node (row).

EXIT AND PEXIT BASICS

Just as the transmitter model in Figure 1 consists of two main elements, so does its receiver (covered in more detail in our companion paper [1]). These two elements are the LDPC decoder and the CPM soft-input soft-output (SISO) module. These decoder modules take turns iterating back and forth, sharing updated *soft information* with each other, in the form of extrinsic log-likelihood ratios (LLRs). Although a LLR is a rigorous probabilistic concept, it also has a deceptively simple format, where it can be viewed as if it was a noisy BPSK sample from an AWGN channel. Consider a simple BPSK channel

$$Y = X + W \tag{2}$$

with input values $X = \pm\sqrt{E_s}$ and additive noise W with zero mean and variance $N_0/2$. Although it is obvious to say so, when conditioned on X, Y has mean $\pm\sqrt{E_s}$ and variance $N_0/2$.

The LLR of this simple BPSK channel is defined as

$$\lambda(X) \triangleq \ln \frac{P(Y|X = -\sqrt{E_s})}{P(Y|X = +\sqrt{E_s})} = \sqrt{E_s} L_c \cdot Y \tag{3}$$

where $P(Y|X)$ is the conditional PDF of Y given X and $L_c = 4/N_0$. Although (2) and (3) have entirely different meanings—one is a model of a physical channel and the other is a probabilistic

expression—they are remarkably similar. In fact, if we view the far right-hand side of (3) itself as the output of an AWGN channel and condition it on X, then the resulting value is a Gaussian random variable with mean $\pm 4E_s/N_0 = \pm\sigma^2/2$ and variance $8E_s/N_0 = \sigma^2$. Thus, an LLR itself can be interpreted as the output of a binary AWGN channel where the scaling is such that the conditional variance is twice the conditional mean. This allows us to use a single value, σ, to characterize the statistics of the LLR.

Now consider two LLRs that are related to the same X but with independent "noise" components and different values of σ, say σ_1 and σ_2. When such LLRs are *added* (as they are in the LDPC decoder's processing steps), then their conditional means and variances *also add*, which results in a combined σ that is

$$\sigma = \sqrt{\sigma_1^2 + \sigma_2^2} \tag{4}$$

The mutual information (MI) between the binary input random variable X embedded in (3) and the Gaussian distributed output is given by [8]

$$J(\sigma) = 1 - \int_{-\infty}^{+\infty} \frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{(y-\sigma^2/2)^2}{2\sigma^2}} \cdot \log_2\left(1 + e^{-y}\right) \, \mathrm{d}y \tag{5}$$

and this represents the capacity of the binary-input AWGN channel. Although the form of (5) is intimidating, a very simple piecewise approximation of $J(\sigma)$ and its inverse, $J^{-1}(I)$, are given in the Appendix of [7]. This function provides a simple translation between the σ domain and the I (information) domain, and its inverse provides the reverse translation [these functions are as simple to use as, for example, $\cos(\cdot)$ and $\cos^{-1}(\cdot)$].

The PEXIT analysis [3] (and the original EXIT analysis [7] before that) employs a structure that mimics the LDPC decoder itself. However, instead of passing actual received channel samples to the actual LDPC decoder, the PEXIT approach injects an initial amount of *information* into the LDPC model (a simple number in the range $0 \leq I \leq 1$), this translates into an initial value of σ [by a simple call to the function $J^{-1}(I)$]. From there, the values of σ flow from variable nodes to check nodes in an iterative fashion, accumulating in the process by repeated use of simple arithmetic such as in (4). The initial amount of information is related to the SNR, again through the function $J(\sigma)$, and the desired result is that σ accumulates iteratively without bound, which translates to $I \to 1$, i.e. convergence to an error free result. The PEXIT computations are surprisingly simple (repeated calls to $J(\cdot)$, $J^{-1}(\cdot)$, addition, square root, etc.) and remarkably accurate. The structure of the LDPC code can be adjusted, "tweaked," or otherwise optimized until the lowest SNR is found that still results in the information converging to unity. This is the *threshold* SNR.

The PEXIT analysis is flexible so that it allows for protomatrix features such as multiple edges ($\mathcal{B} > 1$), punctured variable nodes, and degree-1 variable nodes; furthermore, the PEXIT analysis can accurately resolve decoding thresholds based on the nuances of the connections between variable and check nodes. A major incompatability with the PEXIT analysis and our current application is that the CPM modulator/demodulator is situated *between* the LDPC encoder/decoder and the AWGN channel. This fundamentally alters the flow of extrinsic information in the system. This issue was addressed in studies such as [4] and requires only that we evaluate the EXIT "characteristic function" of the modulation scheme, which we explain next.

EXIT Characteristic Function for CPM. To illustrate the role the modulation plays in the PEXIT analysis, we start with a simple example. For the binary-input AWGN channel (i.e. BPSK), the *a posteriori* probability, LLR format, can be expressed as

$$\lambda(X|Y) = \lambda(X;\mathrm{I}) + \lambda(X) = \lambda(X;\mathrm{I}) + \sqrt{E_s}L_c \cdot Y \tag{6}$$

where $\lambda(X;\mathrm{I})$ is the *a priori* LLR (we use "I" to denote this "input" LLR). The full *a posteriori* LLR is an important "output" of our analysis, but we also have need for an *extrinsic* version of the *a posteriori* LLR, which is $\lambda(X;\mathrm{O})$ (we use "O" to denote this "output" LLR). To get this version of the output, we simply subtract (or "back out") the *a priori* value, which leaves an extrinsic value of $\lambda(X;\mathrm{O}) = \sqrt{E_s}L_c \cdot Y$, which has a value of $J(\sqrt{8E_s/N_0})$ when expressed in the I domain in units of information. Figure 3 (a) shows a plot of the extrinsic *a posteriori* information curve, $I_{\mathrm{E,\,BPSK}} = J(\sqrt{8E_s/N_0})$, as a function of the *a priori* information, $I_{\mathrm{A,\,BPSK}}$. These curves are given for the three values of E_s/N_0 that result in values of $J(\sqrt{8E_s/N_0})$ of 1/2, 2/3, and 4/5. Because $J(\cdot)$ represents the capacity of the binary-input AWGN channel, these values of E_s/N_0 correspond to the maximum achievable code rates of 1/2, 2/3, and 4/5. That these curves are horizontal lines underscores the fact that, for BPSK, the extrinsic *a posteriori* information is independent of the *a priori* information.

In the case of CPM, it is not possible to compute these quantities in closed form [9–11] and so we resort to numerical evaluation via Monte Carlo simulation. The procedure is depicted in block-diagram form in Figure 2. From an outside perspective, the independent (input) variable is I_{A}, the *a priori* information, which we increment in steps of 0.01 (1/100) through the range $0.0 \leq I_{\mathrm{A}} \leq 1.0$. For each value of I_{A}, we compute (by simulation) a corresponding dependent (output) value of I_{E}, which is the extrinsic *a posteriori* information. The simulation is parameterized by the channel SNR, E_s/N_0, which remains fixed during a full "sweep" of the input I_{A} values. This results in an input–output "transfer characteristic," $T_{E_s/N_0}[I_{\mathrm{A}}]$, for each SNR value. This is simply a LUT that returns a value of I_{E} for a given value of I_{A} at the specified SNR. We use square brackets $[\cdot]$ to denote that the LUT index must be rounded to the nearest 1/100 in order to access the nearest output. We compute these LUTs for as many values of SNR as are needed for the PEXIT algorithm in [3, 4], in E_s/N_0 increments of 0.05 dB.

The inner-workings of this simulation are too detailed to be covered herein due to space limitations. However, we have conducted these simulations for ARTM0, ARTM1, and ARTM2, and the results are shown in Figure 3 for the values of E_s/N_0 that correspond to maximum achievable code rates of 1/2, 2/3, and 4/5. We call particular attention to Figure 3 (c), which shows the EXIT curves for ARTM1. With ARTM1 (SOQPSK-TG), it is possible to (1) turn *off* the differential encoding, and (2) approximate the signal as OQPSK, which in turn is identical to BPSK from a channel capacity perspective [12]. Although the ARTM1 curves in Figure 3 (c) were computed numerically, the curves for differential encoding *off* are identical to the BPSK curves in Figure 3 (a) (aside from minor differences in the E_s/N_0 values needed to achieve the desired code rates). From an *information theory* perspective, these flat curves confirm that ARTM1 (SOQPSK-TG) can be made to behave exactly like BPSK. Therefore, it is "easy" to apply BPSK-designed LDPC codes to the case of SOQPSK-TG, which is exactly what was done in IRIG-106. Specifically, the family of BPSK-optimized AR4JA codes [5] were adopted into the IRIG-106 standard for SOQPSK-TG.

A final observation regarding Figure 3 is that the EXIT transfer characteristics have a positive slope and join the point $(1, 1)$ (except for ARTM1 with differential encoding *off*, as discussed earlier). This means that an increase in *a priori* information supplied to the CPM SISO module (i.e. increasing abscissa) will result in a gain in extrinsic *a posteriori* information produced by the CPM SISO module (i.e. increasing ordinate). This provides an information theoretic justification

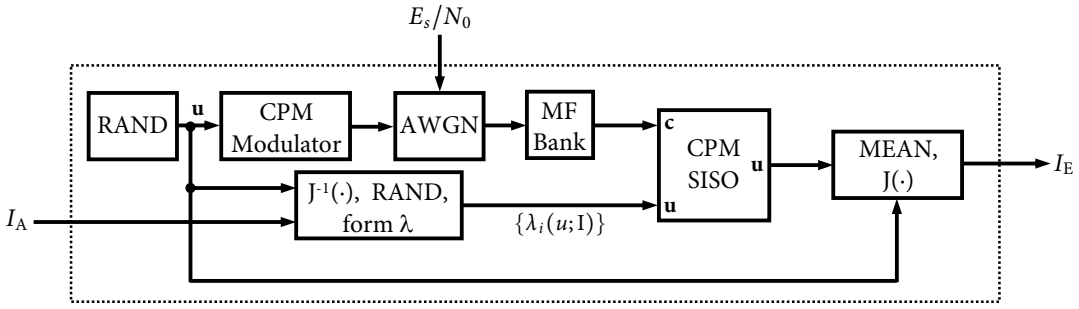

Figure 2: Monte Carlo simulation of CPM EXIT characteristic function.

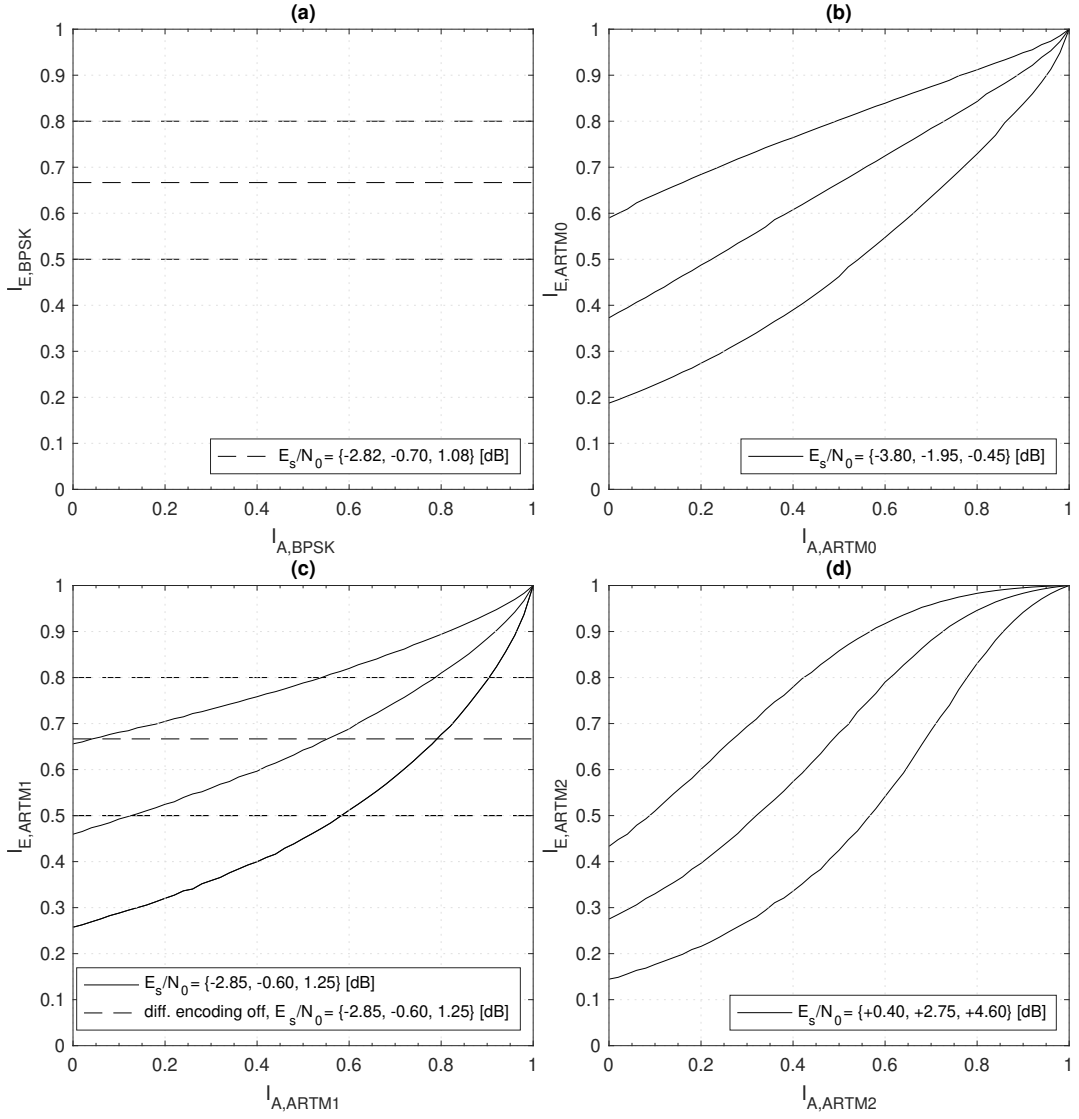

Figure 3: Input–output extrinsic "transfer characteristic," $T_{E_s/N_0}[I_\mathrm{A}]$, for (a) BPSK, (b) ARTM0, (c) ARTM1, and (d) ARTM2. The E_s/N_0 values are indicated in the legend and correspond to the maximum achievable code rates of 1/2, 2/3, and 4/5. For ARTM1 in (c), curves are also given for differential encoding *off*.

for the iterative LDPC/SISO arrangement studied in our companion paper [1], where additional CPM SISO passes using updated *a priori* information from the LDPC decoder lead to performance gains. Conversely, the flat EXIT curves for BPSK (or ARTM1 with differential encoding *off*) mean that additional demodulator passes are pointless because no extrinsic gains are possible.

CPM-Based PEXIT Algorithm. The modifications to the PEXIT algorithm that are necessary for CPM are essentially the same as those that were developed in [4] for partial response channels. The original algorithm in [3] maintains *a priori* ("A") and extrinsic ("E") variables going to/from the variable ("v," indexed by j) and check ("c," indexed by i) nodes that are updated iteratively, with input coming externally from the "channel." In this case, the "channel" is the CPM SISO module ("s"), which itself has an iterative update from *a priori* to extrinsic that involves the EXIT characteristic function for the SNR of the AWGN channel. The primary points where the algorithm needs to be modified are where the "channel" interfaces with the LDPC decoder, because in the present context the "channel" is the CPM SISO module.

In the initialization step, the user specifies E_s/N_0, which in turn specifies the CPM EXIT characteristic function (LUT) $T_{E_s/N_0}[\cdot]$. We set $I_{\text{As}} = 0$, which is the *a priori* information for the SISO module and we enter the PEXIT iterative loop. In the SISO update step, we quantize I_{As} to the nearest available input index in the LUT, $T_{E_s/N_0}[\cdot]$, and then access the LUT to convert the *a priori* information to extrinsic information: $I_{\text{Es}} = T_{E_s/N_0}[I_{\text{As}}]$.

To transfer information to the LDPC decoder, we set $I_{\text{ch}}(j) = I_{\text{Es}}, \forall j$, except we set $I_{\text{ch}}(j) = 0$ if the j-th variable node is punctured. This is the *a priori* information going from the "channel" to the j-th variable node of the LDPC decoder. From there, the PEXIT algorithm proceeds exactly as outlined in [3]. At the end of the PEXIT algorithm, when it is time to transfer information back to the SISO module, the update is given by

$$I_{\text{As}} = \frac{1}{N_{\text{B}}} \sum_{j=0}^{N_{\text{B}}-1} J\left(\sqrt{\sum_{s} b_{s,j} \left[J^{-1}(I_{\text{Av}}(s,j))\right]^2}\right) \tag{7}$$

It is worth pointing out that setting $I_{\text{ch}}(j) = I_{\text{Es}}, \forall j$ implies that the SISO module transfers a uniform amount of information to the variable nodes, which does not vary from node to node (i.e. it is independent of j). Likewise, even though Equation (7) operates on information values from variable nodes that varies with j, they are averaged over j so that a uniform amount of information is transferred back to the SISO module. This assumption of uniformity back and forth between the SISO module and the LDPC decoder in the PEXIT algorithm is akin to the functionality [13] of the interleaver and deinterleaver in the receiver.

LDPC CODE DESIGN FOR CPM

Protomatrix Optimization. In this section, we propose a simple procedure to search for a good protomatrix (protograph) that is "matched" with a specific CPM scheme. The desired properties of the resulting protomatrix are a low decoding threshold and preservation of the linear minimum distance growth property [5], which guarantees no error floor when assigning random circulants in the lifting stage of the design.

To preserve the linear minimum distance growth property, there are several techniques and strategies that can be employed; such as [5]: (a) including some degree-1 variable nodes; (b) including some degree-2 variable nodes via the check-node splitting technique; (c) including some punctured variable nodes; (d) including some high-degree variable nodes; and (e) maintaining a variable node degree of 3 or higher except for the controlled introduction of the degree-1 and

degree-2 nodes mentioned earlier. All of the above techniques were explored in searching for LDPC codes to pair with ARTM0, ARTM1, and ARTM2. We did not see any advantage in decoding threshold by employing degree-1 and degree-2 variable nodes, or by introducing punctured variable nodes. Our best decoding threshold results were obtained by a simple search that focused on variable node degrees between 3 and some upper limit. It is this procedure that we outline next.

We selected a basic protomatrix format with $M_{\mathrm{B}} = 4$ rows and N_{B} columns as required to achieve the desired code rate:

$$\mathbf{B} = \begin{pmatrix} b_{0,0} & b_{0,1} & \cdots & b_{0,N_{\mathrm{B}}-1} \\ b_{1,0} & b_{1,1} & \cdots & b_{1,N_{\mathrm{B}}-1} \\ b_{2,0} & b_{2,1} & \cdots & b_{2,N_{\mathrm{B}}-1} \\ b_{3,0} & b_{3,1} & \cdots & b_{3,N_{\mathrm{B}}-1} \end{pmatrix} \tag{8}$$

This is consistent with starting protomatrix size in [4]. We allow the elements of $\mathbf{B}$ to be drawn from a 4-ary alphabet, i.e. $b_{i,j} \in \{0, 1, 2, 3\}$, with $\mathcal{B} = 4$. The remaining task, while of the utmost importance, is simply to specify the elements of $\mathbf{B}$.

To accomplish this, we optimize the elements of $\mathbf{B}$ on a *column-wise* basis (i.e. on a variable node by variable node basis), which is similar to the approach in [4]. With a 4-ary alphabet and four elements per column, there is an alphabet of $4^4 = 256$ possible columns, with a column sum (column weight, or variable node degree) ranging from 0 to 12. We eliminate any columns with a weight less than three or greater than some upper limit, and we refer to this initial set of candidate columns as $\mathcal{S}_{\text{init}}$. Next, we commence the CPM-based PEXIT algorithm, which was just described. The algorithm must be supplied with a set of EXIT characteristic function curves (LUTs) for the specific CPM scheme that are computed (simulated) over a sufficiently large range of E_s/N_0 values. Figure 3 gives only three such curves for each of the CPMs of interest in this study, but a much larger set of curves was computed in order to support the code design procedure.

A starting value of E_s/N_0, $(E_s/N_0)_{\text{cur}}$, is selected and a full execution of the algorithm takes place for *each* of the candidate columns in $\mathcal{S}_{\text{init}}$ (the remainder of the protomatrix remains constant while the candidate column changes from one PEXIT execution to the next). Any candidate columns that achieve the desired result ($I_{\text{APP}}(j) = 1, \forall j$) are deemed "survivors" and are collected in the set $\mathcal{S}_{\text{cur}}$. The value of $(E_s/N_0)_{\text{cur}}$ is decremented (by 0.05 dB in our case) and the results of the previous step are designated $(E_s/N_0)_{\text{prev}}$ and $\mathcal{S}_{\text{prev}}$. The PEXIT algorithm is again executed once for each of the survivors in $\mathcal{S}_{\text{prev}}$, which produces a new list of survivors, $\mathcal{S}_{\text{cur}}$, that belong to $(E_s/N_0)_{\text{cur}}$. This continues until $(E_s/N_0)_{\text{cur}}$ is decremented to the point where $\mathcal{S}_{\text{cur}}$ is empty (i.e. no survivors), whereupon $(E_s/N_0)_{\text{prev}}$ and $\mathcal{S}_{\text{prev}}$ represent the end result of the optimization for the given column. An "optimized" value for the column can be selected from $\mathcal{S}_{\text{prev}}$ and fixed into the protomatrix. The focus then shifts to another column and can begin at $(E_s/N_0)_{\text{prev}}$ using $\mathcal{S}_{\text{init}}$. At some point, each column has been visited enough that no further progress is made and the decoding threshold is determined to be $(E_s/N_0)_* = (E_s/N_0)_{\text{prev}}$. Clearly, this approach does not produce a unique result; however, in repeated runs, the same general statistical results were achieved each time, which is consistent with other studies on LDPC code ensemble design, cf. e.g. [4, 5, 7].

While there may be many final survivors in $\mathcal{S}_{\text{prev}}$ at the end of each column's optimization, each has an associated PEXIT iteration count, and preference can be given for the survivors with the lowest counts. In some cases, a larger-weight survivor is of interest to ensure some diversity in variable node degree. Furthermore, we added a constraint that any *check node degree* should not exceed 32, because a larger check node degree increases the difficulty in assigning random

circulants in the lifting stage of the design for small lifting factors.

There is a surprising variation in the performance of the final code in terms of BER performance and the required number of decoding iterations; this variation was noticeable primarily with the smallest lifting factors (32 and 64 in our study). This problem was addressed simply by trying several different protomatrix candidate designs (which are, again, not unique), and for each design creating several instances of the final (randomly lifted) code. The candidate codes were simulated against each other and the best one was selected. The authors in [14] mention similar head-to-head trials in selecting the final AR4JA codes. Thus, optimizing the selection of the final code over an ensemble of candidates is an integral part of the code design process.

Lifting the Protomatrix to Obtain the Final Code. The analysis in [15] showed certain performance advantages are possessed by the two-stage lifting procedure that was used to obtain the AR4JA codes [5] and similar codes, e.g. [4]. This provides clear motivation for us to follow a similar approach. Initially, we followed the exact strategy with the AR4JA codes, which was to lift the protomatrix by a factor of 4 using the progressive edge growth (PEG) algorithm [16]. This results in an "intermediate" protomatrix, $\mathbf{D}$, with dimensions $M_{\mathrm{D}} = 4M_{\mathrm{B}}$ and $N_{\mathrm{D}} = 4N_{\mathrm{B}}$. Unlike the original protomatrix, which has elements drawn from the 4-ary alphabet $\{0, 1, 2, 3\}$, $\mathbf{D}$ has elements drawn from the binary alphabet $\{0, 1\}$. In other words, when expressed as *protographs*, the original protograph was allowed to have multiple parallel edges (protomatrix elements greater than 1), but lifting by a factor of 4 results in an intermediate protograph where multiple parallel edges are eliminated.

The downside of this strategy is that it results in relatively small lifting factors in the second and final stage that follows, which limits minimum distance growth [5]. For example, the AR4JA codes begin with a protomatrix with fewer rows than (8), which allows for a larger lifting factor at the end. In our case, lifting by a factor of 4 in the first stage means that the lifting factors in the second step can be only half as large as the AR4JA codes. Because of this, we explored the idea of lifting only by a factor of 2 in the first stage, which allows the lifting factors in the second step to be doubled, thus matching those of the AR4JA codes. Lifting by a factor of 2 means that each element of $\mathbf{B}$ is replaced by a 2×2 sub-matrix, as follows:

$$0 \to \begin{pmatrix} 0 & 0 \\ 0 & 0 \end{pmatrix} \qquad 1 \to \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix} \qquad 2 \to \begin{pmatrix} 1 & 1 \\ 1 & 1 \end{pmatrix} \qquad 3 \to \begin{pmatrix} 2 & 1 \\ 1 & 2 \end{pmatrix} \quad (9)$$

In the last case above, lifting a value of $b_{i,j} = 3$ results in protomatrix elements that remain greater than 1, and thus the final parity check matrix will have parallel edges. However, relative to lifting by a factor of 4, we observed no cases over modulation type, block size, and code rate, where the performance was degraded when lifting by a factor of 2. In fact, in many cases the performance was improved due to the doubling of the final lifting factor.

In the second and final lifting stage, $\mathbf{D}$ is lifted by a factor of M_{L} by assigning random phases to $M_{\mathrm{L}} \times M_{\mathrm{L}}$ circulants using the ACE algorithm [6]. A circulant is a square matrix where each row is a right-hand circular (barrel) shift of the row above. As such, the entire circulant can be specified by providing only the first row; the "weight" of the circulant is the sum of the first row. Because of our first lifting stage, the intermediate protomatrix contains entries drawn only from the alphabet $\{0, 1, 2\}$, and so this second lifting stage results in weight-2, weight-1, or weight-0 (all-zeros) circulants. We simplify the discussion by considering a weight-2 circulant to be the superposition of two weight-1 circulants. A weight-1 circulant can be specified simply by its "phase," which is the location of the non-zero element in the first row and is an integer in the set $\{0, 1, \ldots, M_{\mathrm{L}} - 1\}$.

Table 1: Dimensions and Coding Gains (at BER = 10^{-8}) of the Final LDPC Codes.

K	R	N	M_{L}	M_{B}	N_{B}	M_{D}	N_{D}	Δ_0	Δ_1	Δ_2
1024	4/5	1280	32	4	20	8	40	7.6	7.9	7.0
1024	2/3	1536	64	4	12	8	24	8.6	9.1	8.6
1024	1/2	2048	128	4	8	8	16	9.1	9.9	9.3
4096	4/5	5120	128	4	20	8	40	8.7	9.1	8.1
4096	2/3	6144	256	4	12	8	24	9.5	10.1	9.5
4096	1/2	8192	512	4	8	8	16	9.9	10.8	10.2

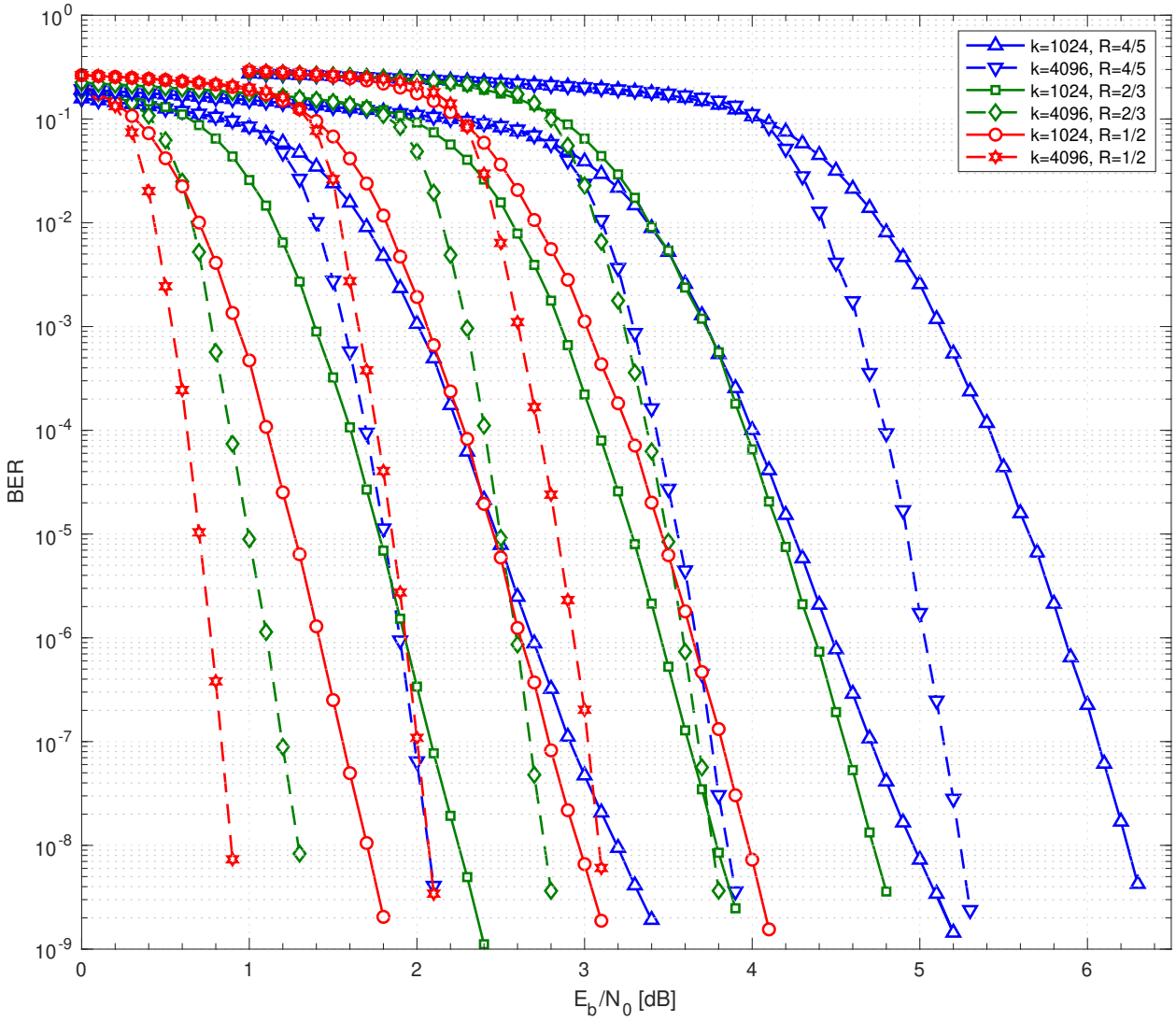

Figure 4: BER curves for the six block sizes and code rates in Table 1. The legend identifies the six cases, however, each modulation type has its own family of six curves: the ARTM0 set is the left-most, the ARTM2 set is the right-most, and the ARTM1 set is in the middle. The coding gains (relative to the *uncoded* cases at BER = 10^{-8}) are listed in Table 1 as Δ_0, Δ_1, and Δ_2 [in dB] for ARTM0, ARTM1, and ARTM2, respectively.

This second lifting stage results in the final low-density parity check matrix, $\mathbf{H}$, with dimensions $M_{\text{H}} = M_{\text{L}}M_{\text{D}}$ and $N_{\text{GH}} = M_{\text{L}}N_{\text{D}}$. The companion to this is the generator matrix, $\mathbf{G}$, with dimensions $K = N_{\text{GH}} - M_{\text{H}}$ and N_{GH} (i.e., both $\mathbf{G}$ and $\mathbf{H}$ have the same number of columns, N_{GH}). Given $\mathbf{H}$, the procedure for obtaining $\mathbf{G}$ is provided in [17].

LDPC DESIGN RESULTS

Because of the existing pairing of the AR4JA codes with SOQPSK-TG (with differential encoding turned *off*), as described in the IRIG-106 standard [2], we are interested in designing codes with similar dimensions and rates. Table 1 shows the desired information block sizes, K, and desired code rates, R (six combinations in total), along with a few parameter values of interest that are driven by these selections.

ARTM0. For ARTM0, the SNR values that correspond to the maximum achievable rates of $R^*_{E_s/N_0} \in \{1/2, 2/3, 4/5\}$ are $(E_s/N_0)_{\text{achievable}} \in \{-3.80, -1.95, -0.45\}$ dB. Using the PEXIT-based protomatrix optimization procedure we outlined above, we were able to design protomatrixes for these code rates with decoding thresholds $(E_s/N_0)_* \in \{-2.95, -1.45, +0.05\}$ dB, which is a gap to capacity of $\{0.85, 0.60, 0.55\}$ dB (less than one dB in all cases).

The resulting protomatrixes are shown below. The subscript notation $\mathbf{B}_{0,1/2}$ designates that the CPM scheme is ARTM0 and the code rate is 1/2, and so forth:

$$\mathbf{B}_{0,1/2} = \begin{pmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 1 & 2 \\ 0 & 0 & 0 & 0 & 1 & 1 & 0 & 2 \\ 0 & 0 & 1 & 1 & 0 & 0 & 1 & 2 \\ 3 & 3 & 2 & 2 & 2 & 2 & 1 & 1 \end{pmatrix} \quad (10) \qquad \mathbf{B}_{0,4/5} = \begin{pmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 2 & 0 & 1 & 2 & 2 & 2 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 & 0 & 2 & 2 & 2 \\ 1 & 1 & 1 & 1 & 1 & 1 & 2 & 2 & 2 & 2 & 2 & 2 & 0 & 2 & 0 & 1 & 2 & 1 & 1 & 0 \\ 2 & 2 & 2 & 2 & 2 & 2 & 1 & 1 & 1 & 1 & 1 & 1 & 2 & 0 & 0 & 1 & 1 & 0 & 0 & 1 \end{pmatrix} \quad (11)$$

$$\mathbf{B}_{0a,2/3} = \begin{pmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 1 & 2 & 1 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 1 & 1 & 1 & 2 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 1 & 0 & 0 & 2 & 2 \\ 3 & 3 & 3 & 3 & 2 & 2 & 2 & 3 & 3 & 2 & 0 & 1 \end{pmatrix} \quad (12) \qquad \mathbf{B}_{0b,2/3} = \begin{pmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 2 & 1 & 2 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 & 2 \\ 1 & 1 & 1 & 1 & 1 & 1 & 2 & 0 & 0 & 0 & 0 & 0 \\ 2 & 2 & 2 & 2 & 2 & 2 & 1 & 2 & 2 & 0 & 1 & 0 \end{pmatrix} \quad (13)$$

For the rate-2/3 case, the protomatrix in (12), $\mathbf{B}_{0a,2/3}$, resulted in excellent BER performance when lifted for the longer information block length of $K = 4096$, but resulted in a much shallower BER slope at low BERs (i.e. an "error floor") for the shorter information block length of $K = 1024$. This was the case regardless of lifting by 2 or 4 in the first lifting stage. With additional trial and error, for $K = 1024$ we selected the protograph in (13), $\mathbf{B}_{0b,2/3}$, which has a slightly worse decoding threshold of $(E_s/N_0)_* = -1.25$ dB but resulted in excellent BER performance.

ARTM1. For ARTM1, the SNR values that correspond to the maximum achievable rates of $(E_s/N_0)_{\text{achievable}} \in \{-2.85, -0.60, +1.25\}$ dB. We were able to design protomatrixes for these code rates with decoding thresholds $(E_s/N_0)_* \in \{-1.90, 0.0, +1.80\}$ dB, which is a gap to capacity of $\{0.95, 0.70, 0.50\}$ dB (again, less than on dB). The resulting protomatrixes are:

$$\mathbf{B}_{1,1/2} = \begin{pmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 1 & 3 \\ 0 & 0 & 0 & 0 & 1 & 1 & 1 & 2 \\ 0 & 0 & 1 & 1 & 1 & 2 & 0 & 1 \\ 3 & 3 & 2 & 2 & 1 & 0 & 1 & 1 \end{pmatrix} \quad (14) \qquad \mathbf{B}_{1,4/5} = \begin{pmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 2 & 1 & 2 & 2 & 2 & 2 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 1 & 0 & 2 & 2 & 2 & 2 \\ 1 & 1 & 1 & 1 & 1 & 1 & 2 & 2 & 2 & 2 & 2 & 2 & 0 & 1 & 0 & 1 & 1 & 1 & 0 & 0 \\ 2 & 2 & 2 & 2 & 2 & 2 & 1 & 1 & 1 & 1 & 1 & 1 & 2 & 1 & 0 & 1 & 0 & 0 & 0 & 1 \end{pmatrix} \quad (15)$$

$$\mathbf{B}_{1a,2/3} = \begin{pmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 1 & 3 \\ 0 & 0 & 0 & 0 & 1 & 1 & 0 & 1 & 1 & 1 & 2 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 1 & 0 & 0 & 1 & 2 & 3 \\ 3 & 3 & 3 & 2 & 2 & 2 & 3 & 3 & 2 & 1 & 1 & 0 \end{pmatrix} \quad (16) \qquad \mathbf{B}_{1b,2/3} = \begin{pmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 2 & 2 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 2 & 2 & 2 \\ 1 & 1 & 1 & 1 & 1 & 1 & 1 & 0 & 1 & 0 & 1 & 2 \\ 2 & 2 & 2 & 2 & 2 & 2 & 2 & 2 & 1 & 0 & 0 & 0 \end{pmatrix} \quad (17)$$

As with the ARTM0 case, we settled on two different protomatrixes for the rate-$2/3$ case for ARTM1, in order to address an error floor for the $K = 1024$ block length. The decoding threshold for $\mathbf{B}_{1b,2/3}$ is slightly degraded at $(E_s/N_0)_* = +0.10$ dB but resulted in excellent BER performance.

By way of comparison, the protomatrixes for the AR4JA codes have decoding thresholds of $(E_s/N_0)_* \in \{-2.40, -0.25, +1.60\}$ dB when applied to ARTM1 with differential encoding turned *off*, which is a smaller gap to capacity of $\{0.45, 0.35, 0.35\}$ dB. It has already been established in [18] that LDPC codes paired with MSK-type CPMs without differential encoding (referred to as "nonrecursive MSK" in [18]) perform better than pairings where differential encoding is present. Thus, the superior performance of the AR4JA codes is to be expected.

ARTM2. For ARTM2, the SNR values that correspond to the maximum achievable rates of $(E_s/N_0)_{\text{achievable}} \in \{+0.40, +2.75, +4.60\}$ dB. We were able to design protomatrixes for these code rates with decoding thresholds $(E_s/N_0)_* \in \{+1.75, +3.60, +5.50\}$ dB, which is a gap to capacity of $\{+1.35, +0.85, +0.90\}$ dB (nearly one dB). No error floors were observed when lifting these protomatrixes to the various block lengths. The resulting protomatrixes are:

$$\mathbf{B}_{2,1/2} = \begin{pmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 1 & 2 \\ 0 & 0 & 0 & 0 & 0 & 1 & 1 & 1 \\ 0 & 0 & 0 & 1 & 2 & 0 & 1 & 0 \\ 3 & 3 & 3 & 2 & 1 & 2 & 1 & 1 \end{pmatrix} \quad (18) \qquad \mathbf{B}_{2,2/3} = \begin{pmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 1 & 2 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 1 & 1 & 1 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 1 & 0 & 0 & 2 \\ 3 & 3 & 3 & 3 & 3 & 2 & 2 & 2 & 3 & 3 & 2 & 0 \end{pmatrix} \quad (19)$$

$$\mathbf{B}_{2,4/5} = \begin{pmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 2 & 1 & 2 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 & 1 & 3 \\ 0 & 0 & 0 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 2 & 2 & 2 & 2 & 1 & 1 & 2 & 1 & 2 & 0 \\ 3 & 3 & 3 & 2 & 2 & 2 & 2 & 2 & 2 & 2 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 0 & 0 & 0 \end{pmatrix} \quad (20)$$

BER PERFORMANCE AND CONCLUSION

Figure 4 shows bit error rate (BER) simulations for the 18 LDPC codes (six codes times three modulations) that were produced in this study and Table 1 lists their coding gains, which range from 7.0 to 10.8 dB! The coding gains are denoted as Δ_0, Δ_1, and Δ_2 [in dB] for ARTM0, ARTM1, and ARTM2, respectively, and use the *uncoded* BER = 10^{-8} crossing points of $E_b/N_0 \in \{10.8, 12.9, 13.3\}$ dB as a reference. The gains are comparable across the code and modulation types, which validates the consistency of the design approach. Furthermore, our decoder in [1] has a high enough throughput to simulate down to very low BERs. Overall, these LDPC codes paired with ARTM0 (PCM/FM) and ARTM 2 (ARTM CPM) can be considered to fill in the "missing" LDPC coding options in the current version of IRIG-106.

In future work, we plan to refine and finalize a specific LDPC code for each of the above cases, i.e. 18 pairs of $\mathbf{H}$ and $\mathbf{G}$, that can be considered for the IRIG-106 standard. Final BER results and other details (parameter optimizations, etc.) will be published at a later date.

ACKNOWLEDGMENT

This work was supported by the Spectrum Relocation Fund (SRF) under the project "Forward Error Correction Codes for IRIG-106 CPM Waveforms." The author would like to thank Kip Temple, Bob Selbrede, and Program Manager Bobbie Wheaton.

REFERENCES

[1] E. Perrins, "LDPC codes for IRIG-106 waveforms: Part II–Receiver design," in *Proc. Int. Telemetering Conf.*, (Las Vegas, NV), Oct. 2023.

[2] Range Commanders Council Telemetry Group, Range Commanders Council, White Sands Missile Range, New Mexico, *IRIG Standard 106-2022: Telemetry Standards*, 2022. (Available on-line at https://www.irig106.org).

[3] G. Liva and M. Chiani, "Protograph LDPC codes design based on EXIT analysis," in *Proc. IEEE Global Telecommun. Conf.*, (Washington, DC), Nov. 2007.

[4] A. N. T. V. Nguyen and D. Divsalar, "Protograph-based LDPC codes for partial response channels," in *Proc. IEEE Int. Conf. Commun.*, (Ottawa, Canada), Jun. 2012.

[5] D. Divsalar, S. Dolinar, C. R. Jones, and K. Andrews, "Capacity-approaching protograph codes," *IEEE J. Select. Areas Commun.*, vol. 27, pp. 876–888, Aug. 2009.

[6] T. Tian, C. R. Jones, J. Villasenor, and R. D. Wesel, "Selective avoidance of cycles in irregular LDPC code construction," *IEEE Trans. Commun.*, vol. 52, pp. 1242–1247, Aug. 2004.

[7] S. ten Brink, G. Kramer, and A. Ashikhmin, "Design of low-density parity-check codes for modulation and detection," *IEEE Trans. Commun.*, vol. 52, pp. 670–678, Apr. 2004.

[8] S. ten Brink, "Convergence behavior of iteratively decoded parallel concatenated codes," *IEEE Trans. Commun.*, vol. 49, pp. 1727–1737, Oct. 2001.

[9] E. Perrins and M. Rice, "Reduced complexity detectors for multi-h CPM in aeronautical telemetry," *IEEE Trans. Aerosp. Electron. Syst.*, vol. 43, pp. 286–300, Jan. 2007.

[10] E. Perrins and M. Rice, "Optimal and reduced complexity receivers for M-ary multi-h CPM," in *Proc. IEEE Wireless Commun. Netw. Conf.*, (Atlanta, Georgia), pp. 1165–1170, Mar. 2004.

[11] P. Chandran and E. Perrins, "Symbol timing recovery for CPM with correlated data symbols," *IEEE Trans. Commun.*, vol. 57, pp. 1265–1270, May 2009.

[12] J. Proakis and M. Salehi, *Digital Communications*. New York: McGraw-Hill, 2008.

[13] E. Perrins and M. Rice, "A simple figure of merit for evaluating interleaver depth for the land-mobile satellite channel," *IEEE Trans. Commun.*, vol. 49, pp. 1343–1353, Aug. 2001.

[14] K. S. Andrews, D. Divsalar, S. Dolinar, J. Hamkins, C. R. Jones, and F. Pollara, "The development of turbo and LDPC codes for deep-space applications," *Proc. IEEE*, vol. 95, pp. 2142–2156, Nov. 2007.

[15] B. K. Butler and P. H. Siegel, "Bounds on the minimum distance of punctured quasi-cyclic LDPC codes," *IEEE Trans. Inform. Theory*, vol. 59, pp. 4584–4597, Jul. 2013.

[16] X.-Y. Hu, E. Eleftheriou, and D.-M. Arnold, "Progressive edge-growth Tanner graphs," in *Proc. IEEE Global Telecommun. Conf.*, (San Antonio, TX, USA), Nov. 2001.

[17] Consultive Committee for Space Data Systems (CCSDS), "Low density parity check codes for use in near-Earth and deep space applications (131.1-O-2 Orange Book)," Sep. 2007.

[18] K. R. Narayanan, İ. Altunbas, and R. S. Narayanaswami, "Design of serial concatenated MSK schemes based on density evolution," *IEEE Trans. Commun.*, vol. 51, pp. 1283–1295, Aug. 2003.