Performance of Low-Density Parity-Check Coded Modulation

Jon Hamkins
Jet Propulsion Laboratory
California Institute of Technology
4800 Oak Grove Drive
Pasadena, CA 91109-8099
Jon.Hamkins@jpl.nasa.gov

Abstract— This paper reports the simulated peformance of each of the nine accumulate-repeat-4-jagged-accumulate (AR4JA) low-density parity-check (LDPC) codes [3] when used in conjunction with binary phase-shift-keying (BPSK), quadrature PSK (QPSK), 8-PSK, 16-ary amplitude PSK (16-APSK), and 32-APSK. We also report the performance under various mappings of bits to modulation symbols, 16-APSK and 32-APSK ring scalings, log-likelihood ratio (LLR) approximations, and decoder variations. One of the simple and well-performing LLR approximations can be expressed in a general equation that applies to all of the modulation types.

TABLE OF CONTENTS

1 INTRODUCTION 1
2 SIGNAL MODEL 2
3 MODULATIONS 2
4 BIT TO SYMBOL MAPPINGS 3
5 LOG LIKELIHOOD RATIO 4
6 NUMERICAL RESULTS 6
7 CONCLUSIONS 6
ACKNOWLEDGEMENTS 6
REFERENCES 9

1. INTRODUCTION

This paper presents no new error correction code construction, modulation type, demodulation method, or decoding technique. Its contribution lies in reporting the simulated performance of many combinations of coded modulations that, as far as we know, have not been previously reported. It also provides a semi-tutorial presentation of the complex baseband representation of various modulation types, various bit-to-modulation-symbol mappings, and the derivation of their assocaited log likelihood ratios (LLRs).

The performance of accumulate-repeat-4-jagged-accumulate (AR4JA) low-density parity-check (LDPC) codes on a binary-input additive white Gaussian noise (AWGN) chanel is well-documented [2], [3]. Such published performance results are for with binary phase-shift keying (BPSK) or quadrature PSK (QPSK), as is typical of most deep space missions. When bandwidth is constrained, however, system engineers may also desire to know the performance of LDPC codes when used with higher order modulations, in order that they may most effectively trade off power efficiency, bandwidth efficiency, and complexity.

The performance of a code when used with a nonbinary modulation may be approximated from its BPSK performance by its code imperfectness. First, the code imperfectness of the code when used with BPSK is determined by measuring the difference between the code's required bit signal to noise ratio E_b/N_0 to attain a given word error probability (P_w) and the minimum possible E_b/N_0 required to attain the same P_w as implied by the sphere-packing bounds for codes with the same block size k and code rate r [5]. This same imperfectness is then applied with respect to the capacity of the higher order modulation to arrive at an approximated performance of the code when used with the higher order modulation. The imperfectness approximation has generally been found to be fairly accurate, to within about 0.5 dB, over a wide variety of codes and modulations.

This paper reports the simulated peformance of the nine AR4JA LDPC codes when used in conjunction with binary phase-shift-keying (BPSK), quadrature PSK (QPSK), 8-PSK, 16-ary amplitude PSK (16-APSK), and 32-APSK. The results are consistent with previously reported performance of rate 4/5 AR4JA codes used with BPSK, 8-PSK, and 16-APSK [4]. Results are included for varying bit-to-symbol mappings used by the modulator and varying log-likelihood ratio (LLR) approximations used by the decoder. One of the simple and well-performing LLR approximations can be expressed in a general equation that applies to all of the modulation types.

This software, written separately in C and Matlab as standalone packages with equivalent functionality, implements encoders and decoders for a set of nine error correcting codes and modulators and demodulators for five modulation types. The software can be used as a single program to simulate the performance of such coded modulation.

The error correcting codes implemented are the nine accumulate repeat-4 jagged accumulate (AR4JA) low-density parity-check (LDPC) codes, which have been approved for international standardization by the Consultative Committee for

[1] 978-1-4244-3888-4/10/$25.00 ©2010 IEEE.
[2] IEEEAC paper # 1244, Version 1, Updated October 25, 2009

Space Data Systems, and which are scheduled to fly on a series of NASA missions in the Constellation Program. The software implements the encoder and decoder functions, and contains compressed versions of generator and parity-check matrices used in these operations.

The software support the modulations of binary phase-shift keying (BPSK), quadrature PSK (QPSK), 8-PSK, 16 amplitude PSK (16-APSK), and 32-APSK. For each modulation type, the software modulator support for various bit-to-modulation-symbol mappings, including the natural order, the Gray code, the anti-Gray code, and the ordering specified by the Digital Video Broadcast Satellite Second Generation standard for 16-APSK and 32-APSK. The software supports hard and soft demodulation, and when soft, it supports both an exact log likelihood computation and an approximate log likelihood computation based on nearest neighbors. The goal of ranging is to accurately determine the distance between a spacecraft and a ground antenna as a function of time. This is accomplished by measuring the time an electromagnetic signal takes to travel between the spacecraft and the ground. By properly accounting for ground and spacecraft processing delays—carefully measured ahead of time in a calibration process—the round-trip light time can be determined.

2. SIGNAL MODEL

To isolate the coded modulation performance from other effects, this paper assumes an additive white Gaussian noise (AWGN) channel with no Doppler, fading, or other channel impairments, no amplifier distortions, and perfect receiver synchronization.

We begin with a general passband signal of the form

$$s(t) = a(t)\cos(2\pi f_c t + \theta(t)) \tag{1}$$

where f_c is the carrier frequency in Hz, and $a(t)$ and $\theta(t)$ are arbitrary modulation-dependent signals. We may rewrite this as

$$s(t) = \text{Re}\left\{\tilde{s}(t)e^{j2\pi f_c t}\right\} \tag{2}$$

where $\tilde{s}(t) = a(t)e^{j\theta(t)}$ is the complex baseband representation of $s(t)$. We may also write $\tilde{s}(t)$ as

$$\tilde{s}(t) = \sqrt{P_c} + \tilde{m}(t) \tag{3}$$

where $\sqrt{P_c}$ is an unmodulated residual carrier signal with complex baseband power P_c, and $\tilde{m}(t)$ is a complex baseband modulation with complex baseband power $P_d = \lim_{T\to\infty}\frac{1}{T}\int_0^T m^2(t)dt$. This can be put back in passband notation using (2), from which the residual carrier signal term $\sqrt{P_c}\cos(2\pi f_c t)$ is readily apparent. The modulations considered in this paper have the form

$$\tilde{m}(t) = \sum_{i=-\infty}^{\infty} m[i]p(t - iT) \tag{4}$$

where $m[i]$ is a member of a signal constellation $m[i] \in C = \{c(0), c(1), \ldots, c(M-1)\}$ in the complex plane, and where $p(t)$ is square pulse shape of symbol duration T:

$$p(t) = \begin{cases} 1 & \text{if } 0 \le t \le T \\ 0 & \text{otherwise.} \end{cases} \tag{5}$$

For our purposes, the residual carrier signal can be assumed to have been filtered out of the modulated received signal or, equivalently, $P_c = 0$. Thus, the received modulated complex baseband signal is of the form

$$\tilde{r}(t) = \tilde{m}(t) + \tilde{n}(t) \tag{6}$$

where $\tilde{n}(t)$ is a complex baseband Gaussian noise with one-sided power-spectral density N_0 in each dimension. At the receiver, $\tilde{r}(t)$ is put through a perfect matched filter, which results in complex soft symbols

$$r[i] = m[i] + n[i] \tag{7}$$

where $n[i]]$, is a complex Gaussian random variable with variance variance $\sigma^2 = 2N_0$ in each of its real and imaginary components.

3. MODULATIONS

In this section, we enumerate the modulation types consiered in this paper, along with their associated complex signal constellations, default orders, and average complex baseband energy.

BPSK is a real-valued constellation with two signal points: $c(0) = A$ and $c(1) = -A$, where A is a scaling factor. This is shown in Fig. 1(a). The average complex baseband symbol energy is $E_s = E[c(i)^2] = A^2$.

QPSK is a complex constellation with four signal points, with $c(i) = \sqrt{2}A\exp\left[j\frac{\pi}{2}\left(i+\frac{1}{2}\right)\right]$, for $i = 0, 1, 2, 3$. The average symbol energy is $E_s = E[c(i)^2] = 2A^2$, double that of BPSK, but with equal energy per transmitted bit.

8-PSK has constellation points $c(i) = A\exp\left[j\frac{\pi}{4}\left(i+\frac{1}{2}\right)\right]$, for $i = 0, 1, \ldots 7$. In general, M-PSK has constellation points $c(i) = A\exp\left[j\frac{2\pi}{M}\left(i+\frac{1}{2}\right)\right]$, for $i = 0, 1, \ldots M-1$. The average symbol energy is $E_s = E[c(i)^2] = A^2$.

16-APSK is a standard of the second generation Digital Video Broadcast for Satellites [1]. It is also referred to as 12/4 APSK or 12/4 QAM. It consists of the union of amplitude-scaled QPSK and 12-PSK signal constellations:

$$c(i) = \begin{cases} r_1\exp\left[j\frac{\pi}{2}\left(i+\frac{1}{2}\right)\right] & i = 0, 1, 2, 3 \\ r_2\exp\left[j\frac{\pi}{6}\left(i+\frac{1}{2}\right)\right] & i = 4, 5, \ldots, 15. \end{cases} \tag{8}$$

The DVB-S2 standard defines the ratio $r_2/r_1 = 3.15, 2.85, 2.75, 2.70, 2.60$ and 2.57 for code rates 2/3, 3/4, 4/5, 5/6, 8/9, and 9/10, respectively. The average symbol energy is $E_s = E[c(i)^2] = (r_1^2 + 3r_2^2)/4$.

32-APSK is also a DVB-S2 standard. It is the union of three PSK constellations:

$$c(i) = \begin{cases} r_1 \exp\left[j\frac{\pi}{2}\left(i+\frac{1}{2}\right)\right] & i = 0,1,2,3 \\ r_2 \exp\left[j\frac{\pi}{6}\left(i-4+\frac{1}{2}\right)\right] & i = 4,5,\ldots,15 \\ r_3 \exp\left[\frac{j(i-16)\pi}{8}\right] & i = 16,17,\ldots,31. \end{cases} \quad (9)$$

The DVB-S2 standard defines the ratios $r_2/r_1 = 2.84, 2.72, 2.64, 2.54$, and 2.53, and $r_3/r_1 = 5.27, 4.87, 4.64, 4.33$, and 4.30 for for code rates 3/4, 4/5, 5/6, 8/9, and 9/10, respectively. The average symbol energy is $E_s = E[c(i)^2] = (r_1^2 + 3r_2^2 + 4r_3^2)/8$.

4. BIT TO SYMBOL MAPPINGS

Encoded bits are assigned to a sequence of corresponding complex constellation points, or modulation symbols. Each of the modulations considered in this paper has a number of constellation points that is a power of two, which makes such bit to symbol mappnigs straightforward.

The signal constellations in the previous section define a natural binary ordering. For example, the 8-PSK constellation points indexed by i =0, 1, 2, 3, 4, 5, 6, and 7 would correspond to the 3-bit patterns 000, 001, 010, 011, 100, 101, 110, and 111, respectively. We refer to this as the natural bit to symbol mapping for the modulation.

Other mappings, such as Gray codes,[3] can often give better performance. There are many Gray codes with the defining property that adjacent members in the list differ in exactly one bit in their binary representation, some with slightly different performance than others. In our simulations, we used the the binary reflected Gray code, which has recently been proven as the optimal mapping for M-PSK modulations. The binary reflected Gray code of length M is obtained from the binary reflected Gray code of length $M/2$ by listing the members $0, 1, \ldots, M-1$, each preceded by a zero, followed by the members $M-1, M-2, \ldots, 0$, each preceded by a one.

The binary relected Gray code has the prefix property, i.e., a length M' Gray code's members are equal to the first M' members of a Gray code of length M, $M > M'$. Thus, when conducting simulations of Gray codes of various lengths, only the longest Gray code need be stored.

An anti-Gray code has the property that adjacent members in the list differ either in all their bits or in all but one of their bits. An anti-Gray code of length M can be obtained from a binary reflected Gray code of length M by removing the last $M/2$ entries and inserting after each of the remaining $M/2$ entries the ones complement of that entry. Anti-Gray codes do not have a prefix property, meaning a separate mapping

[3]Technically, a Gray code is more properly referred to as a Gray *labeling*. A code's performance is not dependent on the order of indexing, whereas with a Gray labeling, the whole point is that it is defined in a particular order. Nevertheless, we use the term Gray codes here, for consistency with common usage.

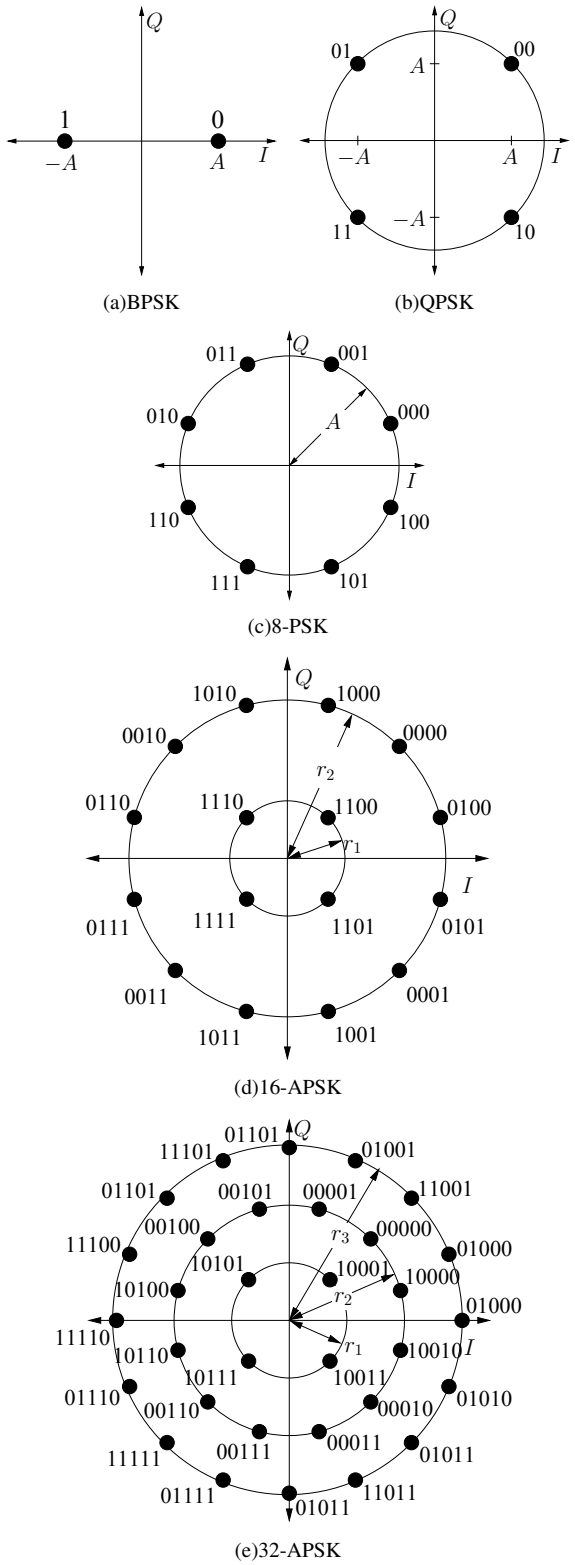

Figure 1. Signal constellations and Gray code bit representations for (a) BPSK, (b) QPSK, and (c) 8-PSK, and DVB-S2 bit representations for (d) 16-APSK and (e) 32-APSK.

Table 1. Bit representations of constellation points. Entries in the table have been converted to decimal.

Natural	Gray	Anti-Gray					DVB-16	DVB-32
0	0	0	0	0	0	0	12	17
1	1	1	3	7	15	31	14	21
2	3		1	1	1	1	15	23
3	2		2	6	14	30	13	19
4	6			3	3	3	4	16
5	7			4	12	28	0	0
6	5			2	2	2	8	1
7	4			5	13	29	10	5
8	12				6	6	2	4
9	13				9	25	6	20
10	15				7	7	7	22
11	14				8	24	3	6
12	10				5	5	11	7
13	11				10	26	9	3
14	9				4	4	1	2
15	8				11	27	5	18
16	24					12		24
17	25					19		8
18	27					13		25
19	26					18		9
20	30					15		13
21	31					16		29
22	29					14		12
23	28					17		28
24	20					10		30
25	21					21		14
26	23					11		31
27	22					20		15
28	18					9		11
29	19					22		27
30	17					8		10
31	16					23		26

should be stored for each length.

For constellations in which constallation points have more than two near neighbors, a specialized bit to symbol mapping is needed. The DVB-S2 standard specifies such a mapping to use with 16-APSK and 32-APSK.

The bit representations of the consellation points under the natural, Gray, anti-Gray, and DVB mappings are given in Table 1, for lengths 2, 4, 8, 16, and 32. Note that in the Gray column, 0, 1, 3, 2, ... in binary is 00000, 00001, 00011, 00010, ..., and each subsequent constellation point has a binary representation that differs in exactly one bit, including wrapping around to the beginning. The anti-Gray column has a separate specification for each length and, for example, 0, 7, 1, 6, ..., in binary is 000, 111, 001, 110, ..., with each entry differing in either two or all three bits.

Table 1 gives a mapping from the constellation index i to the bit representation $map(i)$, but at the modulator we need the inverse operation, to map bits to a constellation point. The inverse is defined by $c_m[map(i)] = c(i)$ for each i, where the subscript m indicates that the constellation has been mapped to a new ordering. For example, to map "1000" to a constellation point using the Gray code, we note that "1000" is 8 in decimal, and $c_m[8] = c(15)$ is the corresponding constellation point.

5. LOG LIKELIHOOD RATIO

Soft-decision decoders take as input the log likelihood ratio LLR for each code bit. Suppose bits $\mathbf{b} = b_{m-1}b_{m-2}\cdots b_0$ are mapped to the complex constellation point $c = c(\mathbf{b})$. Here, we have dropped the subscript m for notational convenience, and assume the $c()$ itself specifies that correct order of symbols for the desired mapping. Let $r = c+n$ denote the noisy received symbol.

Exact LLR

The LLR for the jth bit of the symbol is

$$\begin{aligned}\lambda_j &= \ln\left[\frac{P(b_j=0|r)}{P(b_j=1|r)}\right] \\ &= \ln\left[\frac{p(r|b_j=0)P(b_j=0)/p(r)}{p(r|b_j=1)P(b_j=1)/p(r)}\right] \\ &= \ln\left[\frac{p(r|b_j=0)}{p(r|b_j=1)}\right] \qquad (10)\end{aligned}$$

where we use P to indicate a probability and p to indicate a probability density function (pdf), we applied Bayes theorem for a mixture of probabilities and pdfs, and in the last step we assume $P(b_j = 0) = P(b_j = 1) = 1/2$. For $i \in \{0, 1\}$, we have

$$\begin{aligned}p(r|b_j=i) &= \sum_{\mathbf{b}:b_j=i} p(r|\mathbf{b}) \qquad (11)\\ &= \sum_{\mathbf{b}:b_j=i} p(r-c(\mathbf{b})) \qquad (12)\\ &= \sum_{\mathbf{b}:b_j=i} \frac{1}{2\pi\sigma^2}\exp\left(\frac{-\|r-c(\mathbf{b})\|^2}{2\sigma^2}\right) \qquad (13)\end{aligned}$$

where (11) follows because it is a sum of disjoint events, and (13) is the pdf of a complex Gaussian random variable with variance σ^2 in each of its real and imaginary components. Substituting into (10), we have

$$\lambda_j = \ln\left[\frac{\sum_{\mathbf{b}:b_j=0}\exp\left(\frac{-\|r-c(\mathbf{b})\|^2}{2\sigma^2}\right)}{\sum_{\mathbf{b}:b_j=1}\exp\left(\frac{-\|r-c(\mathbf{b})\|^2}{2\sigma^2}\right)}\right] \qquad (14)$$

Thus, to compute the jth bit LLR from r, one may compute the squared distance to *each* of the constellation points, separating those constellation points that have a 0 in bit j from those that have a 1, and using (14).

We may use the relation

$$\|r-c\|^2 = \|r\|^2 - 2 < r, c > + \|c\|^2 \tag{15}$$

in (14), where the inner product is $\langle r, c\rangle == \text{Re}\{r\}\times\text{Re}\{c\} + \text{Im}\{r\} \times \text{Im}\{c\}$. When the modulation has symbols each of the same energy, as is the case for PSK modulations, the $\|r\|^2$ and $\|c\|^2$ terms in the numerator and denomintor cancel we arrive at the simpler form

$$\lambda_j = \ln\left[\frac{\sum_{\mathbf{b}:b_j=0}\exp\left(\frac{\langle r,c(\mathbf{b})\rangle}{\sigma^2}\right)}{\sum_{\mathbf{b}:b_j=1}\exp\left(\frac{\langle r,c(\mathbf{b})\rangle}{\sigma^2}\right)}\right]. \tag{16}$$

Approximate LLR

A common approximation to the LLR is to approximate each sum in (14) by its largest term, i.e., by using only the nearest constellation point that has $b_j = 0$ in the numerator, and the nearest neighbor that has $b_j = 1$ in the denominator. If we denote these nearest neightbor constellation points by

$$c^*(j,i) == c\left(\text{argmin}_{\mathbf{b}:b_j=i}\|r-c(\mathbf{b})\|^2\right), \tag{17}$$

$i \in \{0,1\}$, we may write

$$\begin{aligned}\lambda_j &\approx \ln\left[\frac{\exp\left(\frac{-\|r-c^*(j,0)\|^2}{2\sigma^2}\right)}{\exp\left(\frac{-\|r-c^*(j,1)\|^2}{2\sigma^2}\right)}\right] \qquad (18)\\ &= \frac{1}{2\sigma^2}\left(\|r-c^*(j,1)\|^2 - \|r-c^*(j,0)\|^2\right)\\ &= \frac{1}{2\sigma^2}\left(2\langle r, c^*(j,0)-c^*(j,1)\rangle + \|c^*(j,1)\|^2 \right.\\ &\quad \left. -\|c^*(j,0)\|^2\right)\end{aligned}$$

or, for equal energy signal constellations,

$$\lambda_j \approx \frac{\langle r, c^*(j,0)-c^*(j,1)\rangle}{\sigma^2} \tag{19}$$

This requires one subtraction and two multiplications. The step of dividing by σ^2 can be eliminated if σ remains constant over many symbols, by precomputing $c(i)/\sigma^2$ for each i.

LLR for BPSK

For BPSK modulation there are only two constellation points, and so the expression in (18), and hence (19), is exact. There is only one bit LLR compute, namely, λ_0, with $c^*(0,0) = A$ and $c^*(0,1) = -A$, and the LLR is given by

$$\lambda_0 = \frac{\langle r, c^*(j,0)-c^*(j,1)\rangle}{\sigma^2} = \frac{\langle r, 2A\rangle}{\sigma^2} = \frac{2A\text{Re}\{r\}}{\sigma^2} \tag{20}$$

LLR for QPSK

As can be seen from Fig. 1(b), the least significant bit (lsb) of a Gray coded QPSK modulation depends on $\text{Re}\{r\}$ in exactly the same way as for BPSK. This can be seen mathematically by noting

$$\begin{aligned} c(0) &= A(1+j)\\ c(1) &= A(-1+j)\\ c(2) &= A(1-j)\\ c(3) &= A(-1-j)\end{aligned}$$

and then plugging these into (16), which becomes

$$\lambda_0 = \ln\left[\frac{\exp\left(\frac{\langle r,c(0)\rangle}{\sigma^2}\right)+\exp\left(\frac{\langle r,c(2)\rangle}{\sigma^2}\right)}{\exp\left(\frac{\langle r,c(1)\rangle}{\sigma^2}\right)+\exp\left(\frac{\langle r,c(3)\rangle}{\sigma^2}\right)}\right] = \frac{2A\text{Re}\{r\}}{\sigma^2} \tag{21}$$

which is identical to (20). Following the same procedure for the most significant bit, where now $c(0)$ and $c(1)$ are in the numerator and $c(2)$ and $c(3)$ are in the denominator, the LLR is given by

$$\lambda_1 = \frac{2A\text{Im}\{r\}}{\sigma^2} \tag{22}$$

Note, when the bit to symbol mapping is not a Gray code, the LLR expressions will not simplify to these expressions with independent real and imaginary components for each bit.

LLR for 8-PSK

The three bit LLRs for each 8-PSK symbol can be computed using (16), with four terms each in the numerator and denominator. As there is no apparent simplification of this exact LLR expression, the approximate LLR computation of (19) can be used when a lower complexity computation is needed.

To identify the closest constellation point with a 0 or a 1 in the bit position of interest, one could compute the distances to all eight constellation points. This is unnecessary, however. As can be seen from Fig. 1(b), if we epress r in polar coordinates as $r = \|r\|e^{j\phi}$, the closest constellation point with lsb equal to zero is given by

$$c^*(0,0) = \begin{cases} c(0) & \text{if } 0 \le \phi < \pi/4\\ c(3) & \text{if } 3\pi/4 \le \phi < \pi\\ c(4) & \text{if } \pi \le \phi < 5\pi/4\\ c(7) & \text{if } 7\pi/4 \le \phi < 2\pi\end{cases} \tag{23}$$

This computation requires only comparisons to constants, and no computaton of distances. Similarly,

$$c^*(0,1) = \begin{cases} c(1) & \text{if } \pi/4 \le \phi < \pi/2\\ c(2) & \text{if } \pi/2 \le \phi < 3\pi/4\\ c(5) & \text{if } 5\pi/4 \le \phi < 3\pi/2\\ c(6) & \text{if } 3\pi/2 \le \phi < 7\pi/4\end{cases} \tag{24}$$

These can then be plugged into (19). The LLRs for the other two bits can be computed in a similar fashion.

LLR for 16-APSK

The four bit LLRs for each 16-APSK symbol can be computed using (16), with eight terms each in the numerator and

denominator. As there is no apparent simplification of this exact LLR expression, the approximate LLR computation of (19) can be used when a lower complexity computation is needed.

To identify the closest constellation point with a 0 or a 1 in the bit position of interest, one could compute the distances to all sixteen constellation points. As was the case for 8-PSK, this is unnecessary. Since 16-APSK is simply the union of two PSK modulations, the angle comparison approach used for 8-PSK can be used to identify the closest inner-ring constellation point with a 0 in the bit position of interest, and separately, to identify the closest outer-ring constellation point. Then $\langle r, c\rangle >$ can be computed for each of the two candidate constellation points to find the closer point. This requires computation of a total of four inner products, or eight multiplications, to compute an approximate bit LLR.

A more careful approach can be even more efficient. The Voronoi regions of 16-APSK are polygonal, so a carefully crafted series of comparisons involving $\|r\|$, $\mathrm{Re}\{r\} - \mathrm{Im}\{r\}$, and ϕ can identify $c * (j, i)$ without multiplications. In this way, only comparisons and the one inner product in (19) would need to be computed.

LLR for 32-APSK

The five bit LLRs for each 32-APSK symbol can be computed using (16), with sixteen terms each in the numerator and denominator. As there is no apparent simplification of this exact LLR expression, the approximate LLR computation of (19) can be used when a lower complexity computation is needed. The same approach as for 16-APSK can be used to avoid all but one inner product computation.

To identify the closest constellation point with a 0 or a 1 in the bit position of interest, one could compute the distances to all 32 constellation points. As was the case for 8-PSK, this is unnecessary. Since 16-APSK is simply the union of two PSK modulations, the angle comparison approach used for 8-PSK can be used to identify the closest constellation point with a 0 in the bit position of interest, from among the inner ring only, and separately, from the outer ring only. Then $\langle r, c\rangle >$ can be computed for each of the two candidate constellation points to find the closer. This requires computation of four inner products, or eight multiplications, to compute the approximate LLR.

A more careful approach can be even more efficient. The Voronoi regions of 16-APSK are polygonal, and a carefully crafted series of comparisons involving $\|r\|$, $\mathrm{Re}\{r\} - \mathrm{Im}\{r\}$, and ϕ can identify $c * (j, i)$ without multiplications. In this way, only comparisons and the one inner product in (19) need to be computed.

6. NUMERICAL RESULTS

Figure 2 shows the performance of AR4JA coded BPSK or QPSK on an AWGN channel, demodulated with an exact LLR computation and quantized to 8 bits, and decoded using up to a maximum of 200 iterations. Bit error rate (BER) and codeword error rates (CWER) are shown for codes of input codeword lengths $k = 1024$, $k = 4096$, and $k = 16384$ and rates 1/2 (red), 2/3 (green), and 4/5 (blue). These simulation results are in agreement with those reported elsewhere [3].

The performance of the same codes when used with 8-PSK, 16-APSK, and 32-APSK is reported in Figures 3, 4, 5, respectively. In each case, a soft demodulator is used to compute the exact bit LLRs.

Figure 6 shows the performance of the decoder as a function of the number of iterations. The results shown are for the the $k = 1024$, $r = 1/2$ AR4JA code used with BPSK on an AWGN channel, demodulated with an exact LLR computation quantized to 8 bits, and with a decoder limited to a maximum of 2, 5, 10, 20, 50, 100, and 200 iterations.

Figures 7, 8, and 9 shows the loss when the demodulator uses hard decision decoding. When taking a hard-decision input, the decoder uses as its LLR the quantity $\log[(1-p)/p]$, where $p = Q(\sqrt{2E_s/N_0})$ is the probability the hard decision is incorrect. Note that this requires the SNR to be known; losses would be larger if the SNR were unknown. The results shown are for all nine AR4JA codes used with BPSK on an AWGN channel.

Table 2 shows the encoding and decoding speed of the C simulations, when run on a desktop PC using a 3 GHz Intel Xeon processor. The decoder is an 8-bit message passing decoder that stops iterating when a codeword is found. Because more iterations are needed at lower SNRs, the speed of such a variable iterations decoder is sensitive to the SNR. The speeds reported in the table refer to a decoder operated at the E_b/N_0 shown, which in each case corresponds to operation at CWER$\approx 10^{-4}$ and represents a reasonable lower limit on SNR at which the decoder would be operated in practice.

ACKNOWLEDGMENTS

The author thanks Kenneth Andrews for providing Matlab code for the $k = 1024$, $r = 1/2$ AR4JA LDPC decoder, and Christopher Jones for providing the edge lists for the AR4JA codes.

The research described in this publication was carried out at the Jet Propulsion Laboratory, California Institute of Technology, under a contract with the National Aeronautics and Space Administration.

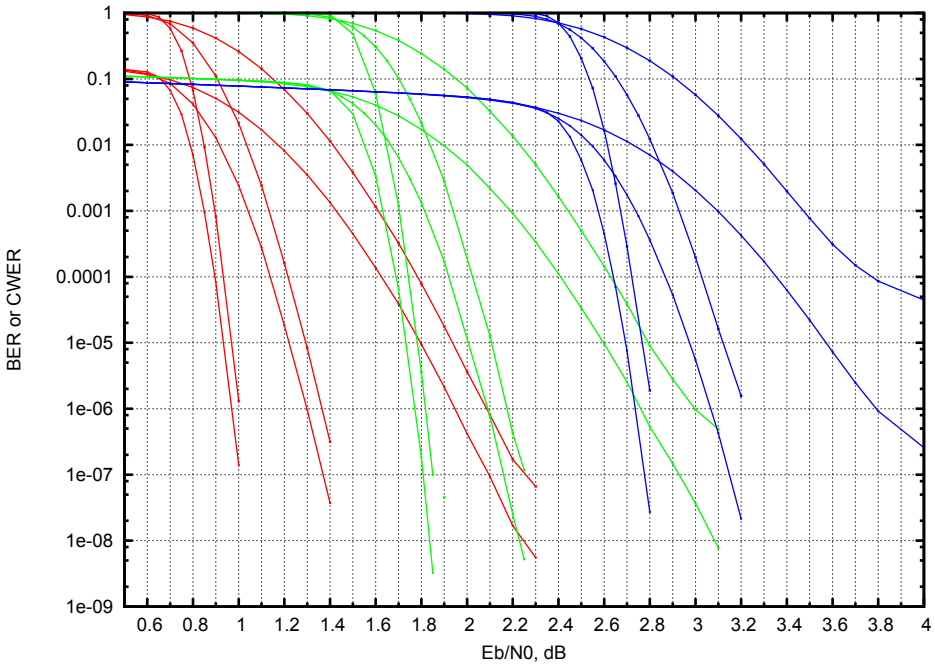

Figure 2. Performance of AR4JA LDPC coded BPSK/QPSK.

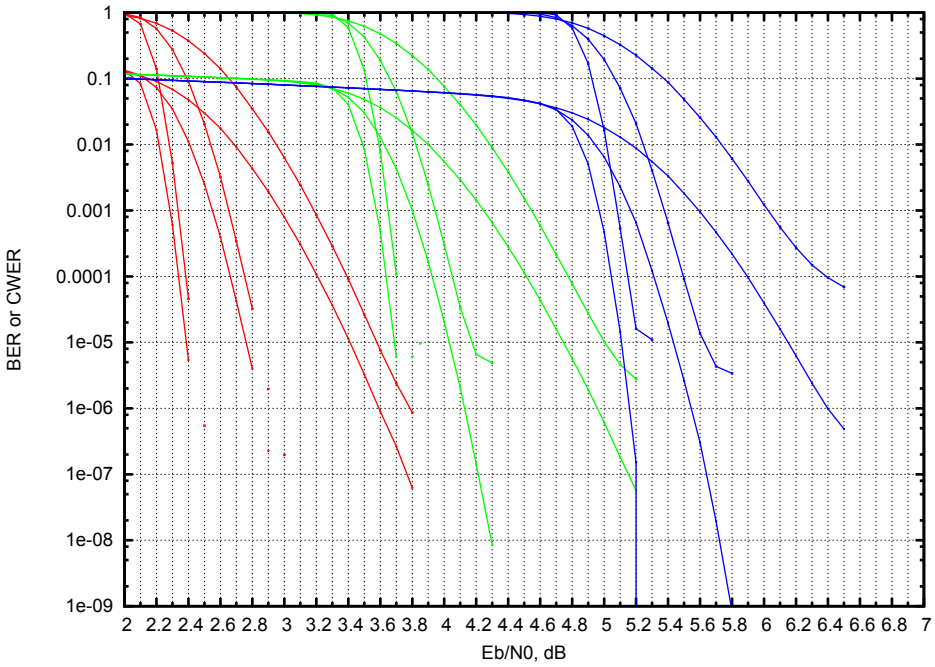

Figure 3. Performance of AR4JA LDPC coded 8-PSK.

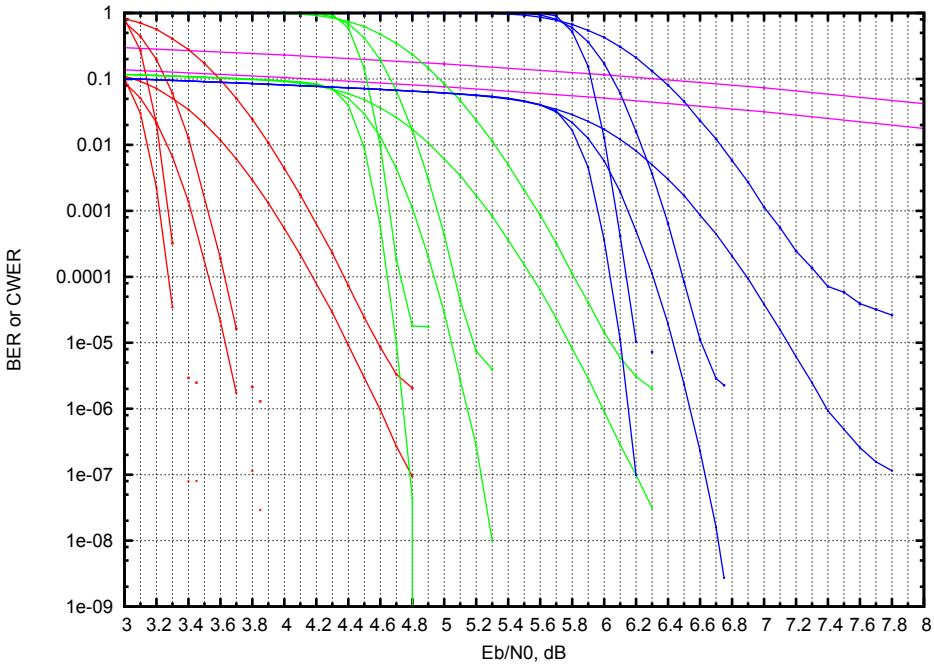

Figure 4. Performance of AR4JA LDPC coded 16-APSK.

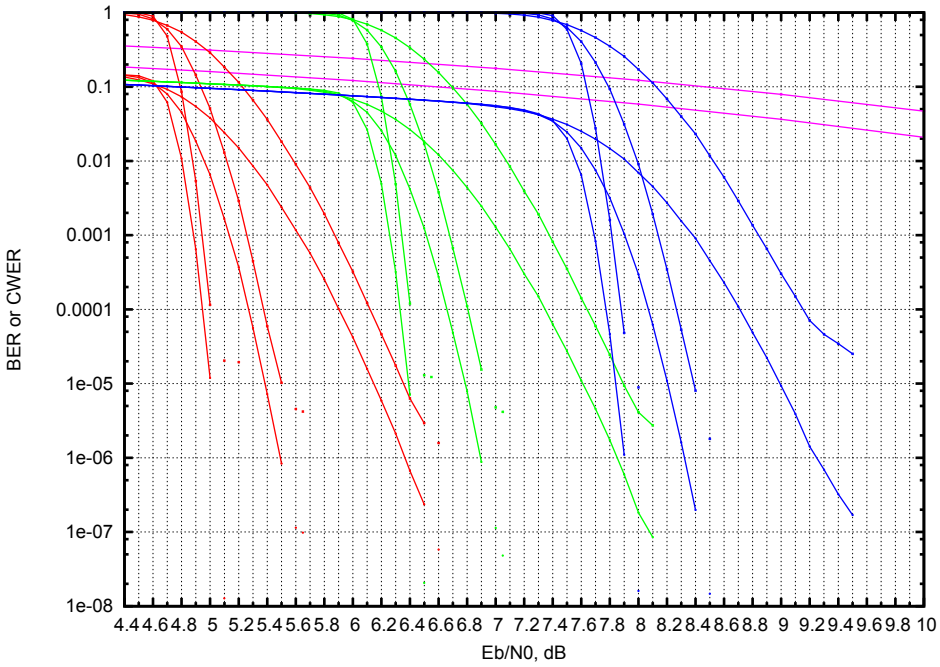

Figure 5. Performance of AR4JA LDPC coded 32-APSK.

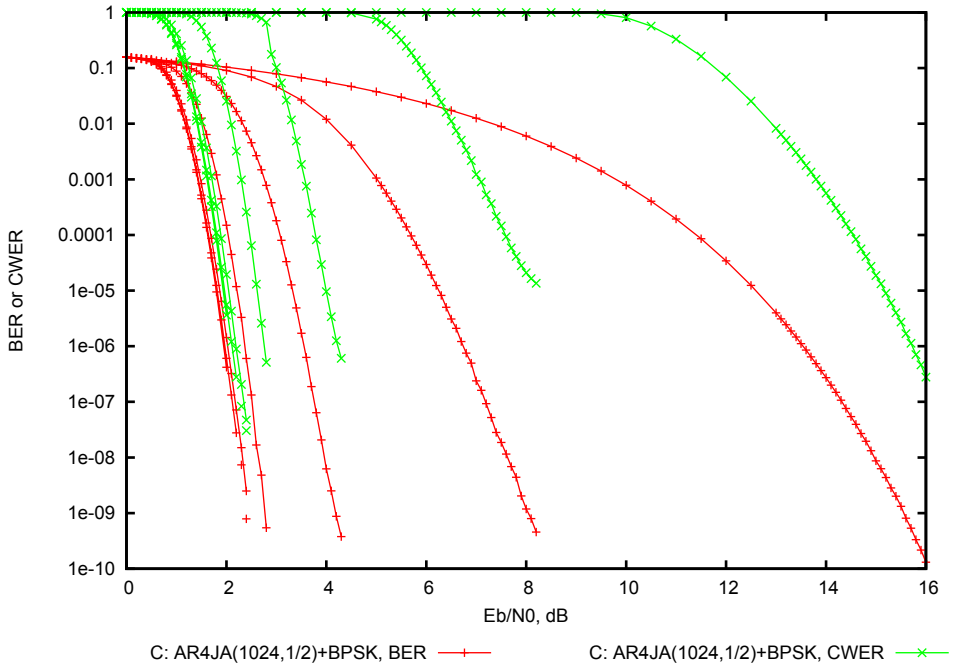

Figure 6. Performance of $k = 1024$, rate 1/2 AR4JA LDPC coded BPSK/QPSK, when decoded with a maximum of 2, 5, 10, 20, 50, 100, and 200 iterations.

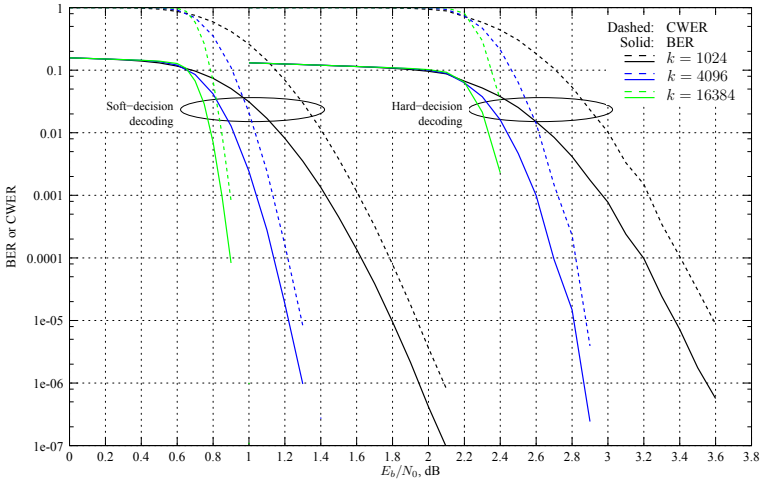

Figure 7. Performance of rate 1/2 AR4JA LDPC coded BPSK/QPSK using hard decision demodulator.

Jon Hamkins received the B.S. degree in electrical engineering from the California Institute of Technology (Caltech), Pasadena, in 1990 and the M.S. and Ph.D. degrees in electrical and computer engineering from the University of Illinois at Urbana-Champaign in 1993 and 1996, respectively. Since then, he has been with the Jet Propulsion Laboratory, Caltech, where he is now supervisor of the Information Processing Group. His research interests include error control theory, information theory, autonomous receivers, ranging, and optical communications.

REFERENCES

[1] ETSI EN 302 307 v1.1.2, "digital video broadcasting (dvb): Second generation framing structure, channel coding and modulation systems for broadcasting, interactive

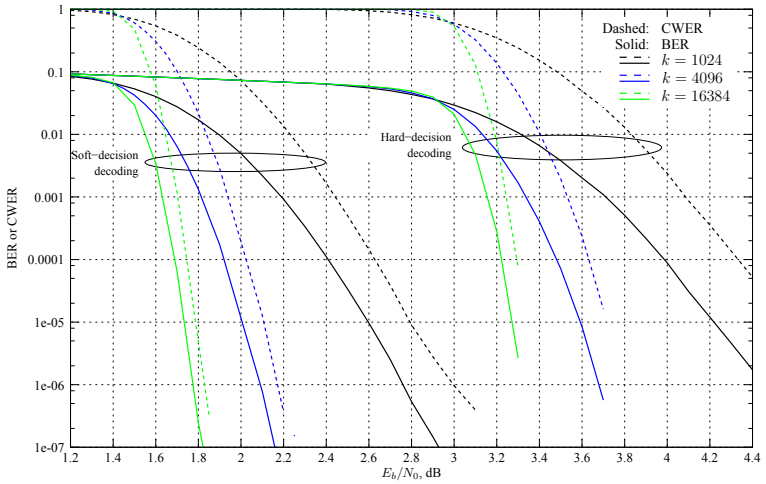

Figure 8. Performance of rate 2/3 AR4JA LDPC coded BPSK/QPSK using hard decision demodulator.

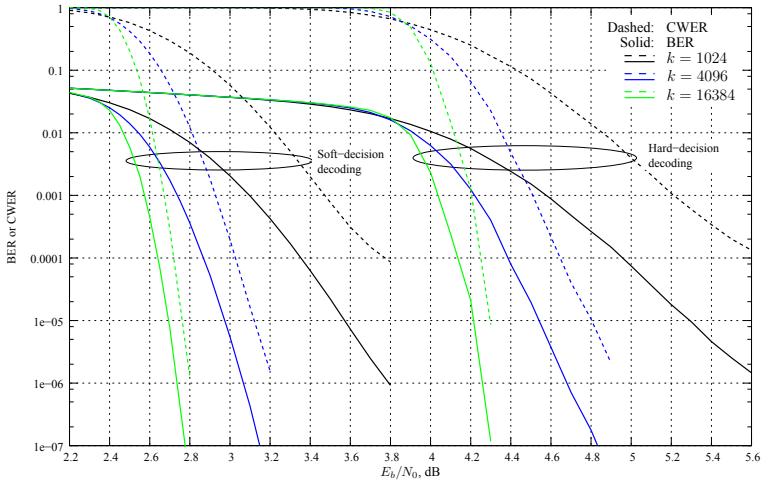

Figure 9. Performance of rate 4/5 AR4JA LDPC coded BPSK/QPSK using hard decision demodulator.

Table 2. Measured speed of encoder and decoder C simulations.

k	rate	E_b/N_0 (dB)	Average iterations	Encoder speed (Mbps)	Decoder speed (Mbps)
1024	1/2	1.8	16.44	14	0.597
1024	2/3	2.6	12.86	25.9	0.928
1024	4/5	3.7	9.2	49.5	1.41
4096	1/2	1.25	27.75	8.23	0.357
4096	2/3	2	22.94	14.4	0.537
4096	4/5	3	16.74	33.8	0.789
16384	1/2	0.95	46.03	1.57	0.219
16384	2/3	1.75	35.11	3.53	0.347
16384	4/5	2.75	23.97	6.45	0.541

services, news gathering and other broadband satellite applications", June 2006.

[2] Low density parity check codes for use in near-earth and deep space. CCSDS 131.1-O-2. Orange Book, Issue 2. September 2007, http://public.ccsds.org/publications/archive/131x1o2e2.pdf.

[3] 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. *Proceedings of the IEEE*, 95(11):2142–2156, Nov. 2007.

[4] Michael K. Cheng, Dariush Divsalar, and Stephanie Duy. Structured low-density parity-check codes with bandwidth efficient modulation. In *Proceedings of SPIE Conference on Defense Security and Sensing*, April 2009.

[5] S. Dolinar, D. Divsalar, and F. Pollara. Code performance as a function of block size. *TDA Progress Report*, 42(133), May 1998.