Turbo Codes for Deep-Space Communications

D. Divsalar and F. Pollara
Communications Systems Research Section

Turbo codes were recently proposed by Berrou, Glavieux, and Thitimajshima [2], and it has been claimed these codes achieve near-Shannon-limit error correction performance with relatively simple component codes and large interleavers. A required E_b/N_o of 0.7 dB was reported for a bit error rate of 10^{-5}, using a rate 1/2 turbo code [2]. However, some important details that are necessary to reproduce these results were omitted. This article confirms the accuracy of these claims, and presents a complete description of an encoder/decoder pair that could be suitable for deep-space applications, where lower rate codes can be used. We describe a new simple method for trellis termination, analyze the effect of interleaver choice on the weight distribution of the code, and introduce the use of unequal rate component codes, which yields better performance.

I. Introduction

Turbo codes were recently proposed by Berrou, Glavieux, and Thitimajshima [2] as a remarkable step forward in high-gain, low-complexity coding. It has been claimed these codes achieve near-Shannon-limit error correction performance with relatively simple component codes and large interleavers. A required E_b/N_0 of 0.7 dB was reported for a bit error rate (BER) of 10^{-5}, using a rate 1/2 turbo code [2]. However, some important details that are necessary to reproduce these results were omitted. The purpose of this article is to shed some light on the accuracy of these claims and to present a complete description of an encoder/decoder pair that could be suitable for deep-space applications, where lower rate codes can be used. Two new contributions are reported in this article: a new, simple method for trellis termination and the use of unequal component codes, which results in better performance.

II. Parallel Concatenation of Convolutional Codes

The codes considered in this article consist of the parallel concatenation of two convolutional codes with a random interleaver between the encoders. Figure 1 illustrates a particular example that will be used in this article to verify the performance of these codes. The encoder contains two recursive binary convolutional encoders, with M_1 and M_2 memory cells, respectively. In general, the two component encoders may not be identical. The first component encoder operates directly on the information bit sequence $\mathbf{u} = (u_1, \cdots, u_N)$ of length N, producing the two output sequences $\mathbf{x}_{1i}$ and $\mathbf{x}_{1p}$. The second component encoder operates on a reordered sequence of information bits, $\mathbf{u}'$, produced by an interleaver of length N, and outputs the two sequences $\mathbf{x}_{2i}$ and $\mathbf{x}_{2p}$. The interleaver is a pseudorandom block scrambler defined by a permutation of N elements with no repetitions: a complete block is read into the interleaver and read out in a specified permuted order. Figure 1 shows an example where a rate $r = 1/n = 1/4$ code is generated by two component codes with $M_1 = M_2 = M = 4$, producing the

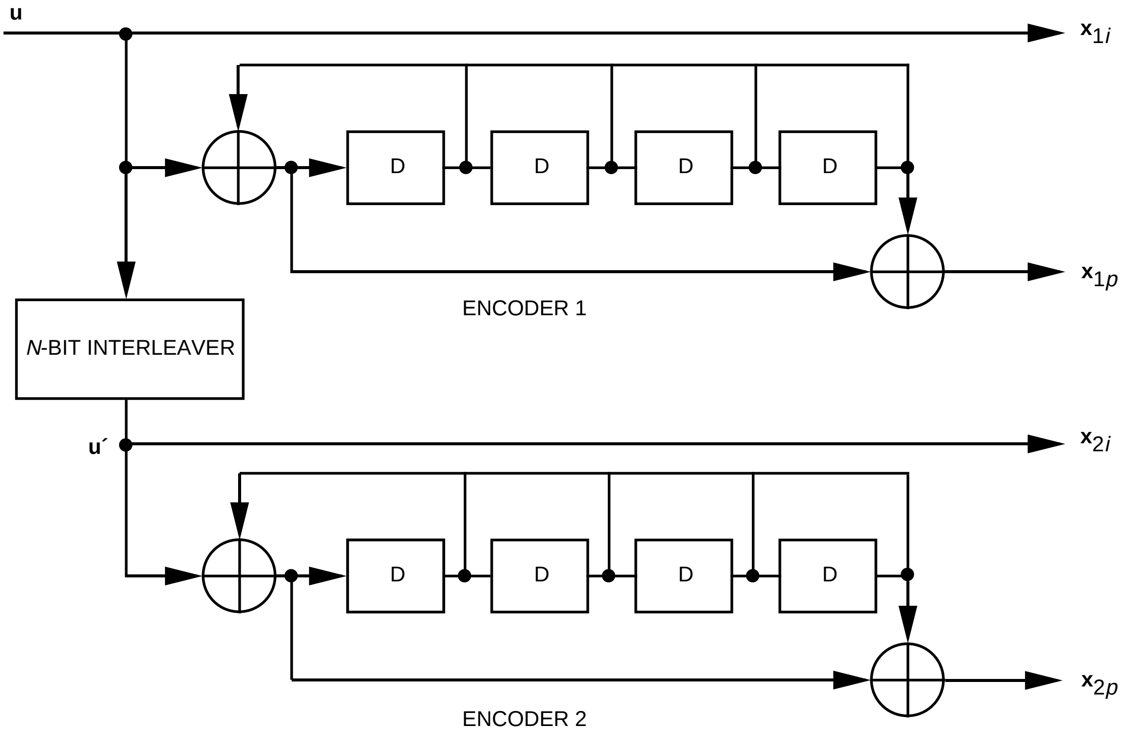

Fig. 1. Example of an encoder.

outputs $\mathbf{x}_{1i} = \mathbf{u}$, $\mathbf{x}_{1p} = \mathbf{u} \cdot g_a/g_b$, $\mathbf{x}_{2i} = \mathbf{u}'$, and $\mathbf{x}_{2p} = \mathbf{u}' \cdot g_a/g_b$, where the generator polynomials g_a and g_b have an octal representation of 21 and 37, respectively. Note that various code rates can be obtained by puncturing the outputs.

A. Trellis Termination

We use the encoder in Fig. 1 to generate a $(n(N+M), N)$ block code. Since the component encoders are recursive, it is not sufficient to set the last M information bits to zero in order to drive the encoder to the all-zero state, i.e., to terminate the trellis. The termination (tail) sequence depends on the state of each component encoder after N bits, which makes it impossible to terminate both component encoders with the same M bits. Fortunately, the simple stratagem illustrated in Fig. 2 is sufficient to terminate the trellis. Here the switch is in position "A" for the first N clock cycles and is in position "B" for M additional cycles, which will flush the encoders with zeros. The decoder does not assume knowledge of the M tail bits. The same termination method can be used for unequal rate and memory encoders.

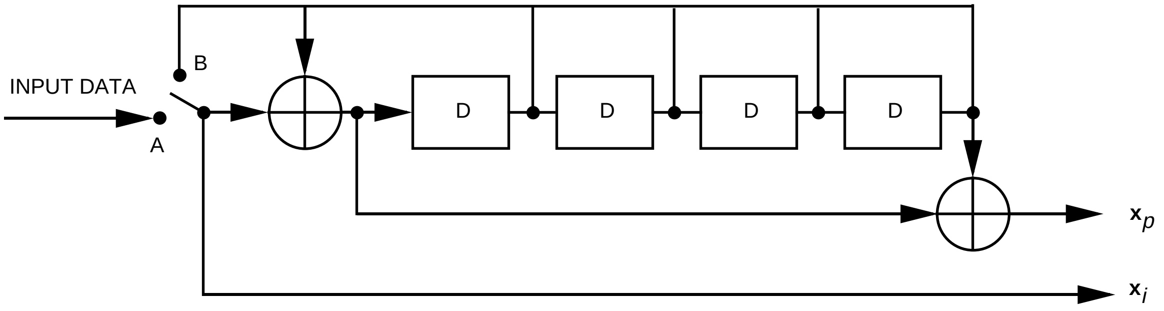

Fig. 2. Trellis termination.

B. Weight Distribution

In order to estimate the performance of a code, it is necessary to have information about its minimum distance, d, weight distribution, or actual code geometry, depending on the accuracy required for the bounds or approximations. The example of turbo code shown in Fig. 1 produces two sets of codewords, $\mathbf{x}_1 = (\mathbf{x}_{1i}, \mathbf{x}_{1p})$ and $\mathbf{x}_2 = (\mathbf{x}_{2i}, \mathbf{x}_{2p})$, whose weights can be easily computed. The challenge is in finding the pairing of codewords from each set, induced by a particular interleaver. Intuitively, we would like to avoid pairing low-weight codewords from one encoder with low-weight words from the other encoder. Many such pairings can be avoided by proper design of the interleaver. However, if the encoders are not recursive, the low-weight codeword generated by the input sequence $\mathbf{u} = (00 \cdots 0000100 \cdots 000)$ with a single "1" will always appear again in the second encoder, for any choice of interleaver. This motivates the use of recursive encoders, where the key ingredient is the recursiveness and not the fact that the encoders are systematic. For our example using a recursive encoder, the input sequence $\mathbf{u} = (00 \cdots 0010000100 \cdots 000)$ generates the minimum weight codeword (weight = 6). If the interleaver does not properly "break" this input pattern, the resulting minimum distance will be 12.

However, the minimum distance is not the most important quantity of the code, except for its asymptotic performance, at very high E_b/N_o. At moderate signal-to-noise ratios (SNRs), the weight distribution at the first several possible weights is necessary to compute the code performance. Estimating the complete weight distribution for a large N is still an open problem for these codes. We have investigated the effect of the interleaver on the weight distribution on a small-scale example where $N = 16$. This yields an (80,16) code whose weight distribution can be found by exhaustive enumeration. Some of our results are shown in Fig. 3(a), where it is apparent that a good choice of the interleaver can increase the minimum distance from 12 to 14, and, more importantly, can reduce the count of codewords at low weights. Figure 3(a) shows the weight distribution obtained by using no interleaver, a reverse permutation, and a 4 × 4 block interleaver, all with $d = 12$. Better weight distributions are obtained by the "random" permutation $\{2, 13, 0, 3, 11, 15, 6, 14, 8, 9, 10, 4, 12, 1, 7, 5\}$ with $d = 12$, and by the best-found permutation $\{12, 3, 14, 15, 13, 11, 1, 5, 6, 0, 9, 7, 4, 2, 10, 8\}$ with $d = 14$. For comparison, the binomial distribution is also shown. The best known (80,16) linear block code has a minimum distance of 28. For an interleaver length of $N = 1024$, we were only able to enumerate all codewords produced by input sequences with weights 1, 2, and 3. This again confirmed the importance of the interleaver choice for reducing the number of low-weight codewords. Better weight distributions were obtained by using "random" permutations than by using structured permutations as block or reverse permutations.

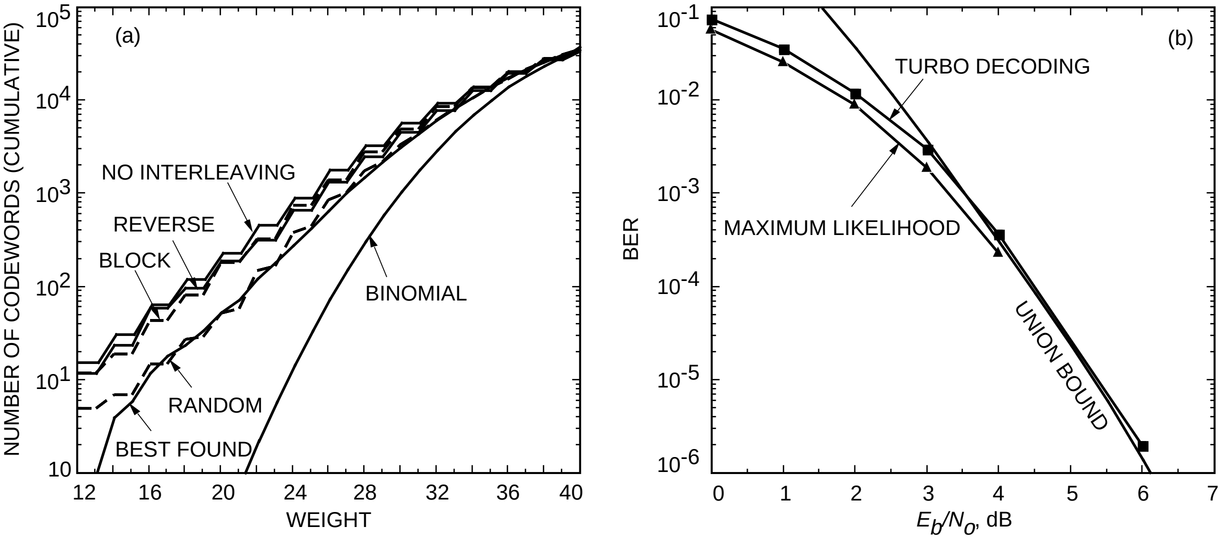

Fig. 3. The (80, 16) code (a) weight distribution and (b) performance.

For the (80,16) code using the best-found permutation, we have compared the performance of a maximum-likelihood decoder (obtained by simulation) to that of a turbo decoder with 10 iterations, as described in Section 3, and to the union bound computed from the weight distribution, as shown in Fig. 3(b). As expected, the performance of the turbo decoder is slightly suboptimum.

III. Turbo Decoding

Let u_k be a binary random variable taking values in $\{+1, -1\}$, representing the sequence of information bits. The maximum a posteriori (MAP) algorithm, summarized in the Appendix, provides the log likelihood ratio $L(k)$ given the received symbols $\mathbf{y}$:

$$L(k) = \log \frac{P(u_k = +1|\mathbf{y})}{P(u_k = -1|\mathbf{y})} \tag{1}$$

The sign of $L(k)$ is an estimate, $\hat{u}_k$, of u_k, and the magnitude $|L(k)|$ is the reliability of this estimate, as suggested in [3].

The channel model is shown in Fig. 4, where the n_{1ik}'s and the n_{1pk}'s are independent identically distributed (i.i.d.) zero-mean Gaussian random variables with unit variance, and $\rho = \sqrt{2E_s/N_o} = \sqrt{2rE_b/N_o}$ is the SNR. A similar model applies for encoder 2.

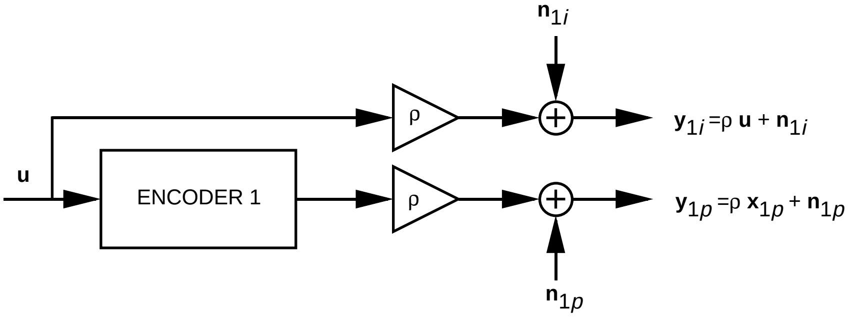
n_{1i}
u
ENCODER 1
ρ
ρ
$y_{1i} = \rho\, u + n_{1i}$
$y_{1p} = \rho\, x_{1p} + n_{1p}$
n_{1p}

Fig. 4. The channel model.

Given the turbo code structure in Fig. 1, the optimum decoding rule maximizes either $P(u_k|\mathbf{y}_1, \mathbf{y}_2)$ (minimum bit-error probability rule) or $P(\mathbf{u}|\mathbf{y}_1, \mathbf{y}_2)$ (maximum-likelihood sequence rule). Since this rule is obviously too complex to compute, we resort to a suboptimum decoding rule [2,3] that separately uses the two observations $\mathbf{y}_1$ and $\mathbf{y}_2$, as shown in Fig. 5. Each decoder in Fig. 5 computes the a posteriori probabilities $P(u_k|\mathbf{y}_i, \tilde{\mathbf{u}}_i), i = 1, 2$ see Fig. 6(a), or equivalently the log-likelihood ratio $L_i(k) = \log\left(P(u_k = +1|\mathbf{y}_i, \tilde{\mathbf{u}}_i)\right) / \left(P(u_k = -1|\mathbf{y}_i, \tilde{\mathbf{u}}_i)\right)$ where $\tilde{\mathbf{u}}_1$ is provided by decoder 2 and $\tilde{\mathbf{u}}_2$ is provided by decoder 1 (see Fig. 6(b)). The quantities $\tilde{\mathbf{u}}_i$ correspond to "new data estimates," "innovations," or "extrinsic information" provided by decoders 1 and 2, which can be used to generate a priori probabilities on the information sequence $\mathbf{u}$ for branch metric computation in each decoder.

The question is how to generate the probabilities $P(\tilde{u}_{i,k}|u_k)$ that should be used for computation of the branch transition probabilities in MAP decoding. It can be shown that the probabilities $P(u_k|\tilde{u}_{i,k})$ or, equivalently, $\log\left(P(u_k = +1|\tilde{u}_{i,k})\right) / \left(P(u_k = -1|\tilde{u}_{i,k})\right)$, $i = 1, 2$ can be used instead of $P(\tilde{u}_{i,k}|u_k)$ for branch metric computations in the decoders. When decoder 1 generates $P(u_k|\tilde{u}_{2,k})$ or $\log\left(P(u_k = +1|\tilde{u}_{2,k})\right) / \left(P(u_k = -1|\tilde{u}_{2,k})\right)$ for decoder 2, this quantity should not include the contribution due to $\tilde{u}_{1,k}$, which has already been generated by decoder 2. Thus, we should have

$$\log \frac{P(u_k = +1|\tilde{u}_{2,k})}{P(u_k = -1|\tilde{u}_{2,k})} = \log \frac{P(u_k = +1|\mathbf{y}_1, \tilde{u}_{1,1}, \cdots, \tilde{u}_{1,k-1}, \tilde{u}_{1,k+1}, \cdots, \tilde{u}_{1,N})}{P(u_k = -1|\mathbf{y}_1, \tilde{u}_{1,1}, \cdots, \tilde{u}_{1,k-1}, \tilde{u}_{1,k+1}, \cdots, \tilde{u}_{1,N})} \tag{2}$$

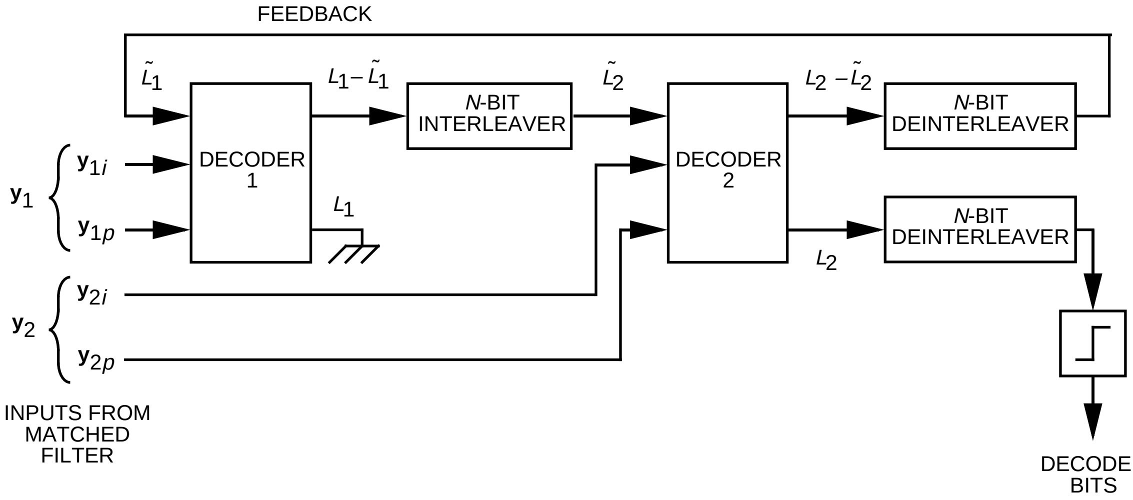

Fig. 5. The turbo decoder.

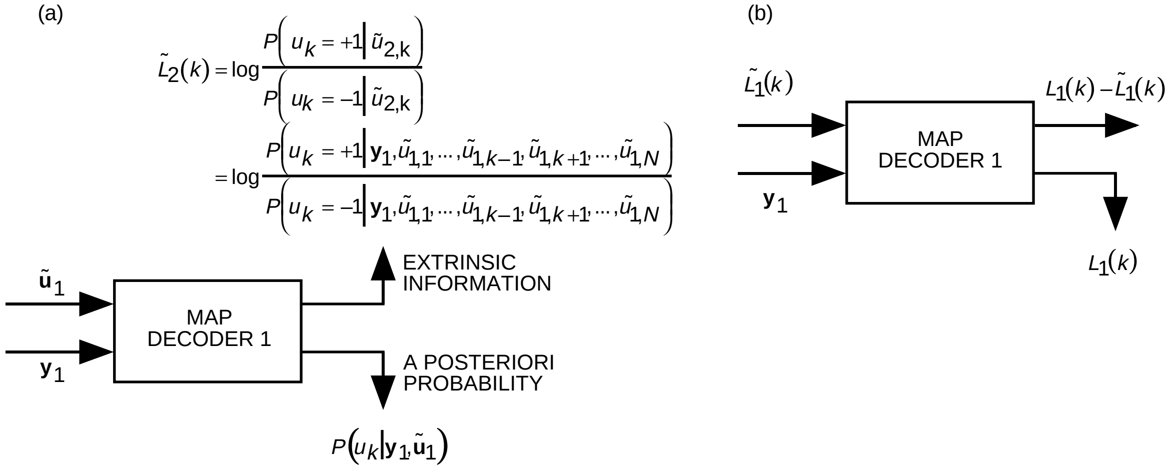
(a)
$\tilde{L}_2(k) = \log \frac{P\left(u_k = +1 \middle| \tilde{u}_{2,k}\right)}{P\left(u_k = -1 \middle| \tilde{u}_{2,k}\right)}$
$= \log \frac{P\left(u_k = +1 \middle| \mathbf{y}_1, \tilde{u}_{1,1}, \cdots, \tilde{u}_{1,k-1}, \tilde{u}_{1,k+1}, \cdots, \tilde{u}_{1,N}\right)}{P\left(u_k = -1 \middle| \mathbf{y}_1, \tilde{u}_{1,1}, \cdots, \tilde{u}_{1,k-1}, \tilde{u}_{1,k+1}, \cdots, \tilde{u}_{1,N}\right)}$
$\tilde{\mathbf{u}}_1$
$\mathbf{y}_1$
MAP DECODER 1
EXTRINSIC INFORMATION
A POSTERIORI PROBABILITY
$P\left(u_k \middle| \mathbf{y}_1, \tilde{\mathbf{u}}_1\right)$
(b)
$\tilde{L}_1(k)$
$\mathbf{y}_1$
MAP DECODER 1
$L_1(k) - \tilde{L}_1(k)$
$L_1(k)$

Fig. 6. Input/output of the MAP decoder: (a) a posteriori probability and (b) log-likelihood ratio.

To compute $\log\left(P(u_k = +1|\tilde{u}_{2,k})\right) / \left(P(u_k = -1|\tilde{u}_{2,k})\right)$, we note [see Fig. 6(a)] that

$$P(u_k|\mathbf{y}_1, \tilde{\mathbf{u}}_1) =$$

$$\frac{P(u_k|\mathbf{y}_1, \tilde{u}_{1,1}, \cdots, \tilde{u}_{1,k-1}, \tilde{u}_{1,k+1}, \cdots, \tilde{u}_{1,N}) P(\tilde{u}_{1,k}|u_k, \mathbf{y}_1, \tilde{u}_{1,1}, \cdots, \tilde{u}_{1,k-1}, \tilde{u}_{1,k+1}, \cdots, \tilde{u}_{1,N})}{P(\tilde{u}_{1,k}|\mathbf{y}_1, \tilde{u}_{1,1}, \cdots, \tilde{u}_{1,k-1}, \tilde{u}_{1,k+1}, \cdots, \tilde{u}_{1,N})} \quad (3)$$

Since $\tilde{u}_{1,k}$ was generated by decoder 2 and deinterleaving is used, this quantity depends only weakly on $\mathbf{y}_1$ and $\tilde{u}_{1,j}$, $j \neq k$. Thus, we can have the following approximation:

$$P(\tilde{u}_{1,k}|u_k, \mathbf{y}_1, \tilde{u}_{1,1}, \cdots, \tilde{u}_{1,k-1}, \tilde{u}_{1,k+1}, \cdots, \tilde{u}_{1,N}) \approx P(\tilde{u}_{1,k}|u_k) = 2P(u_k|\tilde{u}_{1,k})P(\tilde{u}_{1,k}) \quad (4)$$

Using Eq. (4) in Eq. (3), we obtain

$$P(u_k|\mathbf{y}_1, \tilde{u}_{1,1}, \cdots, \tilde{u}_{1,k-1}, \tilde{u}_{1,k+1}, \cdots, \tilde{u}_{1,N}) = \frac{P(u_k|\mathbf{y}_1, \tilde{\mathbf{u}}_1)P(\tilde{u}_{1,k}|\mathbf{y}_1, \tilde{u}_{1,1}, \cdots, \tilde{u}_{1,k-1}, \tilde{u}_{1,k+1}, \cdots, \tilde{u}_{1,N})}{2P(u_k|\tilde{u}_{1,k})P(\tilde{u}_{1,k})} \tag{5}$$

It is preferable to work with likelihood ratios to avoid computing probabilities not involving u_k (see Fig. 6(b)). Define

$$\tilde{L}_i(k) = \log \frac{P(u_k = +1|\tilde{u}_{i,k})}{P(u_k = -1|\tilde{u}_{i,k})}, \quad i = 1, 2 \tag{6}$$

From Eqs. (2) and (5), we obtain $\tilde{L}_2^{(m)}(k) = L_1^{(m)}(k) - \tilde{L}_1^{(m-1)}(k)$ at the output of decoder 1, before interleaving, for the mth iteration. Similarly, we can obtain $\tilde{L}_1^{(m)}(k) = L_2^{(m)}(k) - \tilde{L}_2^{(m)}(k)$ at the output of decoder 2, after deinterleaving. Using the above definitions, the a priori probabilities can be computed as

$$P(u_k = +1|\tilde{u}_{i,k}) = \frac{e^{\tilde{L}_i(k)}}{1 + e^{\tilde{L}_i(k)}} = 1 - P(u_k = -1|\tilde{u}_{i,k}), \quad i = 1, 2 \tag{7}$$

Then the update equation for the mth iteration of the decoder in Fig. 5 becomes

$$\tilde{L}_1^{(m)}(k) = \tilde{L}_1^{(m-1)}(k) + \alpha_m \left[L_2^{(m)}(k) - L_1^{(m)}(k)\right], \quad \alpha_m = 1 \tag{8}$$

This looks like the update equation of a steepest descent method, where $\left[L_2^{(m)}(k) - L_1^{(m)}(k)\right]$ represents the rate of change of $L(k)$ for a given u_k, and α_m is the step size.

Figure 7 shows the probability density function of $\tilde{L}_1(k)$ at the output of the second decoder in Fig. 1, after deinterleaving and given $u_k = +1$. As shown in Fig. 7, this density function shifts to the right as the number of iterations, m, increases. The area under each density function to the left of the origin represents the BER if decoding stops after m iterations.

At this point, certain observations can be made. Note that $\tilde{L}_2(k')$ at the input of decoder 2 includes an additive component $2\rho y_{1ik}$, which contributes to the branch metric computations in decoder 2 at observation y_{2ik}. This improves by 3 dB the SNR of the noisy information symbols at the input of decoder 2. Similar arguments hold for $\tilde{L}_1(k)$. An apparently more powerful decoding structure can be considered, as shown in Fig. 8. However, the performances of the decoding structures in Figs. 8 and 5 are equivalent for a large number of iterations (the actual difference is one-half iteration). If the structure in Fig. 8 is used, then the log-likelihood ratio $\tilde{L}_2(k)$ fed to decoder 2 should not depend on $\tilde{u}_{1k}$ and y'_{1ik}, and, similarly, $\tilde{L}_1(k)$ should not depend on $\tilde{u}_{2k}$ and y'_{2ik}. Using analogous derivations based on Eqs. (2) through (5), we obtain

$$\tilde{L}_2(k) = L_1(k) - \tilde{L}_1(k) - 2\rho y'_{1ik}$$

$$\tilde{L}_1(k) = L_2(k) - \tilde{L}_2(k) - 2\rho y'_{2ik}$$

where $\mathbf{y}'_{1i}$ is the sum of $\mathbf{y}_{1i}$ with the deinterleaved version of $\mathbf{y}_{2i}$ and $\mathbf{y}'_{2i}$ is the sum of $\mathbf{y}_{2i}$ with the interleaved version of $\mathbf{y}_{1i}$. Thus, the net effect of the decoding structure in Fig. 8 is to explicitly pass to decoder 2 the information contained in $\mathbf{y}_{1i}$ (and vice versa), but to remove the identical term from the input log-likelihood ratio.

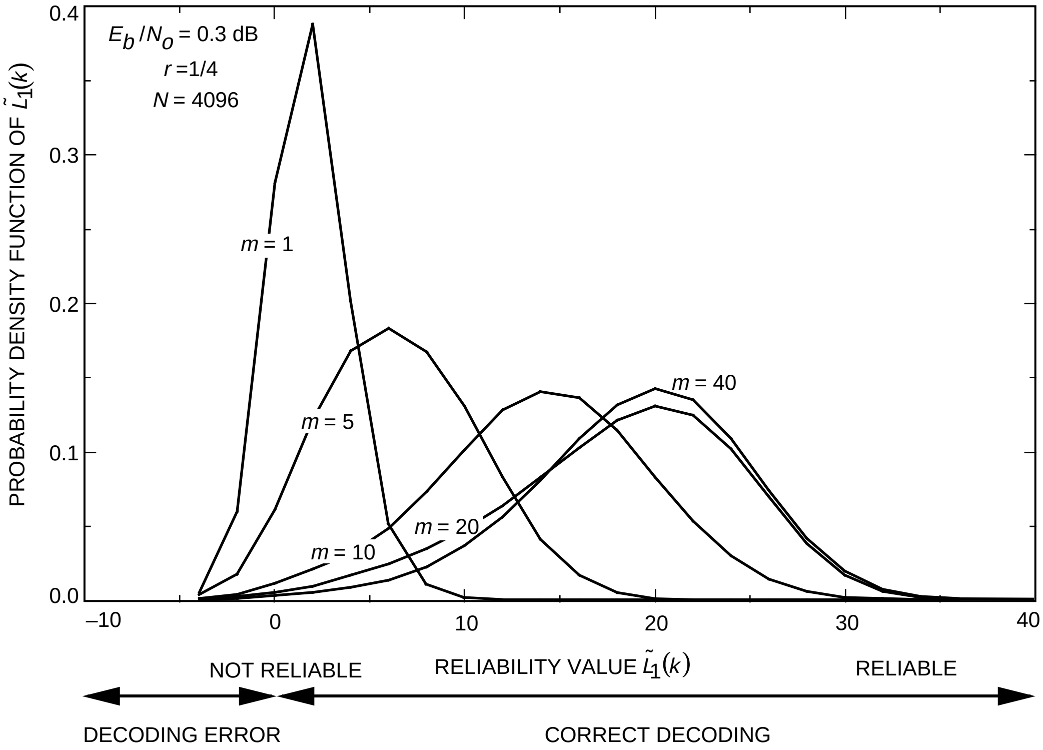

Fig. 7. The reliability function.

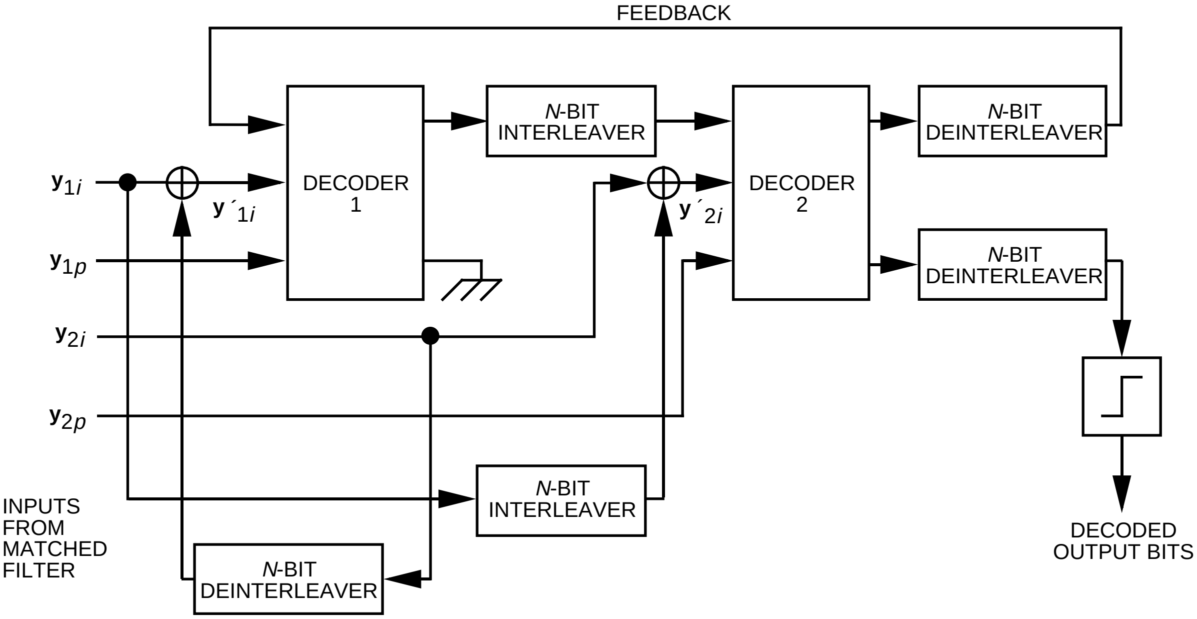

Fig. 8. An equivalent turbo decoder.

IV. Performance

The performance obtained by turbo decoding the code in Fig. 1 with random permutations of lengths $N = 4096$ and $N = 16384$ is compared in Fig. 9 to the capacity of a binary-input Gaussian channel for rate $r = 1/4$ and to the performance of a (15,1/4) convolutional code originally developed at JPL for the Galileo mission. At BER $= 5 \times 10^{-3}$, the turbo code is better than the (15,1/4) code by 0.25 dB for $N = 4096$ and by 0.4 dB for $N = 16384$.

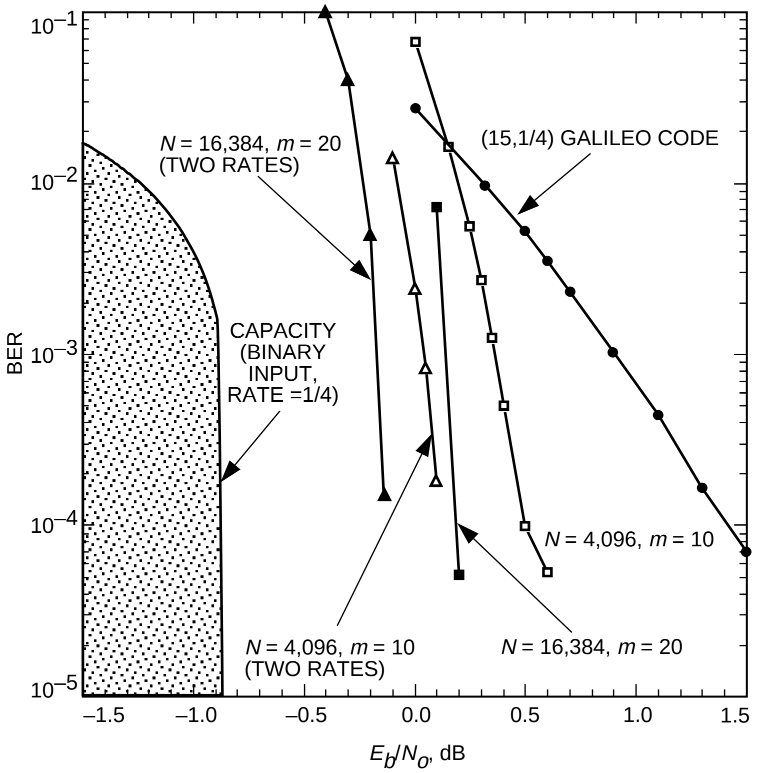

Fig. 9. Turbo codes performance, *r* = 1/4.

So far we have considered only component codes with identical rates, as shown in Fig. 1. Now we propose to extend the results to encoders with unequal rates, as shown in Fig. 10. This structure improves the performance of the overall rate 1/4 code, as shown in Fig. 9. The gains at BER $= 5 \times 10^{-3}$ relative to the (15,1/4) code are 0.55 dB for $N = 4096$ and 0.7 dB for $N = 16384$. For both cases, the performance is within 1 dB of the Shannon limit at BER $= 5 \times 10^{-3}$, and the gap narrows to 0.7 dB for $N = 16384$ at a low BER.

V. Conclusions

We have shown how turbo codes and decoders can be used to improve the coding gain for deep-space communications while decreasing the decoding complexity with respect to the large constraint-length convolutional codes currently in use. These are just preliminary results that require extensive further analysis. In particular, we need to improve our understanding of the influence of the interleaver choice on the code performance, to explore the sensitivity of the decoder performance to the precision with which we can estimate E_b/N_o, and to establish whether there might be a flattening of the performance curves at higher E_b/N_o, as it appears in one of the curves in Fig. 9. An interesting theoretical question is to determine how random these codes can be so as to draw conclusions on their performance based on comparison with random coding bounds.

In this article, we have explored turbo codes using only two encoders, but similar constructions can be used to build multiple-encoder turbo codes and generalize the turbo decoding concept to a truly distributed decoding system where each subdecoder works on a piece of the total observation and tentative estimates are shared among decoders until an acceptable degree of consensus is reached.

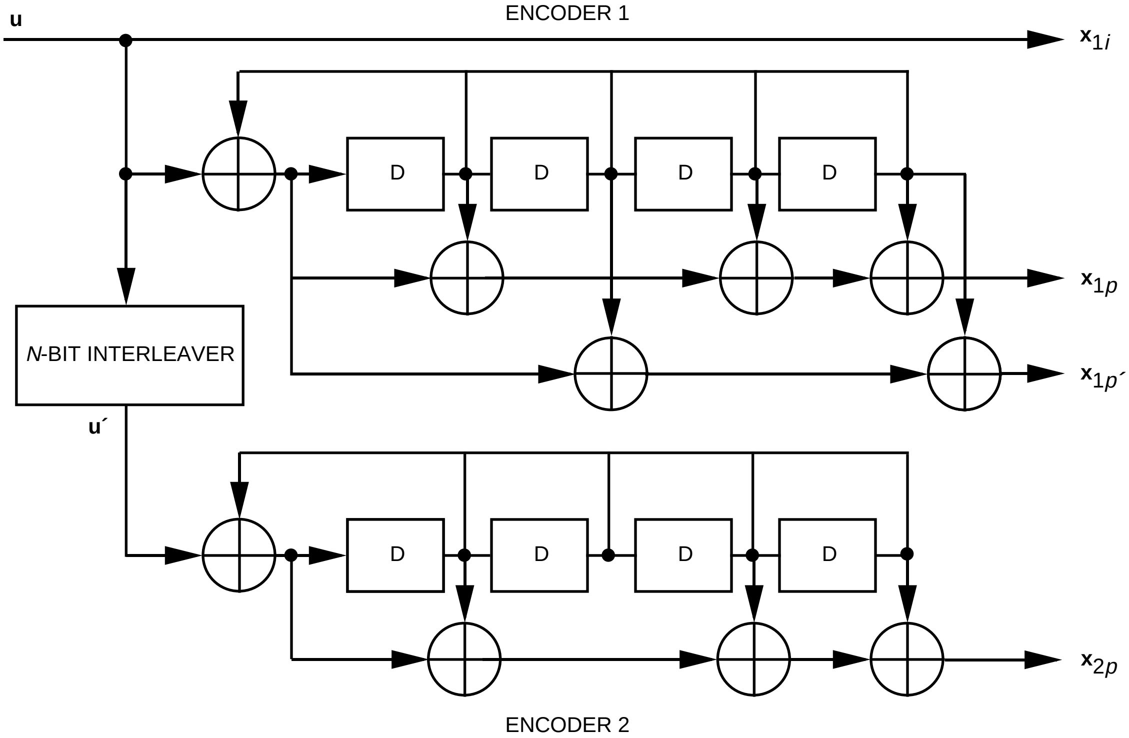

Fig. 10. The two-rate encoder.

Acknowledgments

The authors are grateful to Sam Dolinar and Robert McEliece for their helpful comments.

References

[1] L. Bahl, J. Cocke, F. Jelinek, and J. Raviv, “Optimal Decoding of Linear Codes for Minimizing Symbol Error Rate,” *IEEE Transactions on Information Theory*, vol. 20, pp. 284–287, March 1974.

[2] C. Berrou, A. Glavieux, and P. Thitimajshima, “Near Shannon Limit Error-Correcting Coding and Decoding: Turbo-Codes,” *Proceedings of ICC ’93*, Geneva, Switzerland, pp. 1064–1070, May 1993.

[3] J. Hagenauer and P. Robertson, "Iterative (Turbo) Decoding of Systematic Convolutional Codes With the MAP and SOVA Algorithms," *Proceedings of the ITG Conference on "Source and Channel Coding,"* Frankfurt, Germany, pp. 1-9, October 1994.

[4] P. L. McAdam, L. R. Welch, and C. L. Weber, "M.A.P. Bit Decoding of Convolutional Codes" (Abstract), *1972 International Symposium on Information Theory*, Asilomar, California, p. 91, May 1972.

Appendix
The MAP Algorithm

Let u_k be the information bit associated with the transition from time $k-1$ to time k, and use s as an index for the states. The MAP algorithm [1,4] provides the log likelihood given the received symbols y_k, as shown in Fig. A-1.

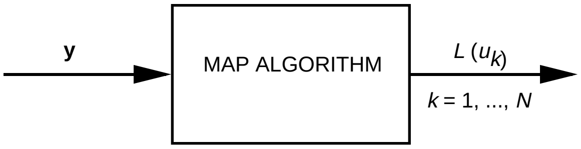

Fig. A-1. The MAP algorithm.

$$L(k) = \log \frac{P(u_k = +1|\mathbf{y})}{P(u_k = -1|\mathbf{y})} = \log \frac{\sum_s \sum_{s'} \gamma_{+1}(y_k, s', s)\alpha_{k-1}(s')\beta_k(s)}{\sum_s \sum_{s'} \gamma_{-1}(y_k, s', s)\alpha_{k-1}(s')\beta_k(s)} \qquad \text{(A-1)}$$

The estimate of the transmitted bits is then given by $sign[L(k)]$ and their reliability by $|L(k)|$. In order to compute Eq. (A-1), we need the *forward* and *backward* recursions,

$$\alpha_k(s) = \frac{\sum_{s'} \sum_{i=\pm1} \gamma_i(y_k, s', s)\alpha_{k-1}(s')}{\sum_s \sum_{s'} \sum_{j=\pm1} \gamma_j(y_k, s', s)\alpha_{k-1}(s')}$$

$$\text{(A-2)}$$

$$\beta_k(s) = \frac{\sum_{s'} \sum_{i=\pm1} \gamma_i(y_{k+1}, s, s')\beta_{k+1}(s')}{\sum_s \sum_{s'} \sum_{j=\pm1} \gamma_j(y_{k+1}, s', s)\alpha_k(s')}$$

where

$$\gamma_i(y_k, s', s) = \begin{cases} \eta_k e^{\rho \sum_{\nu=1}^{n} y_{k,\nu} x_{k,\nu}(s', i)} & \text{if transition } s' \to s \text{ is allowable for } u_k = i \\ 0 & \text{otherwise} \end{cases} \tag{A-3}$$

$\rho = \sqrt{2(E_s/N_0)}$, $\eta_k = P(u_k = \pm 1|\tilde{u}_k)$, except for the first iteration in the first decoder, where $\eta_k = 1/2$, and $x_{k,\nu}$ are code symbols. The operation of these recursions is shown in Fig. A-2. The evaluation of Eq. (A-1) can be organized as follows:

Step 0: $\alpha_0(0) = 1 \quad \alpha_0(s) = 0, \forall s \neq 0$

$\beta_N(0) = 1 \quad \beta_N(s) = 0, \forall s \neq 0$

Step 1: Compute the γ_k's using Eq. (A-3) for each received set of symbols y_k.

Step 2: Compute the α_k's using Eq. (A-2) for $k = 1, \cdots, N$.

Step 3: Use Eq. (A-2) and the results of Steps 1 and 2 to compute the β_k's for $k = N, \cdots, 1$.

Step 4: Compute $L(k)$ using Eq. (A-1) for $k = 1, \cdots, N$.

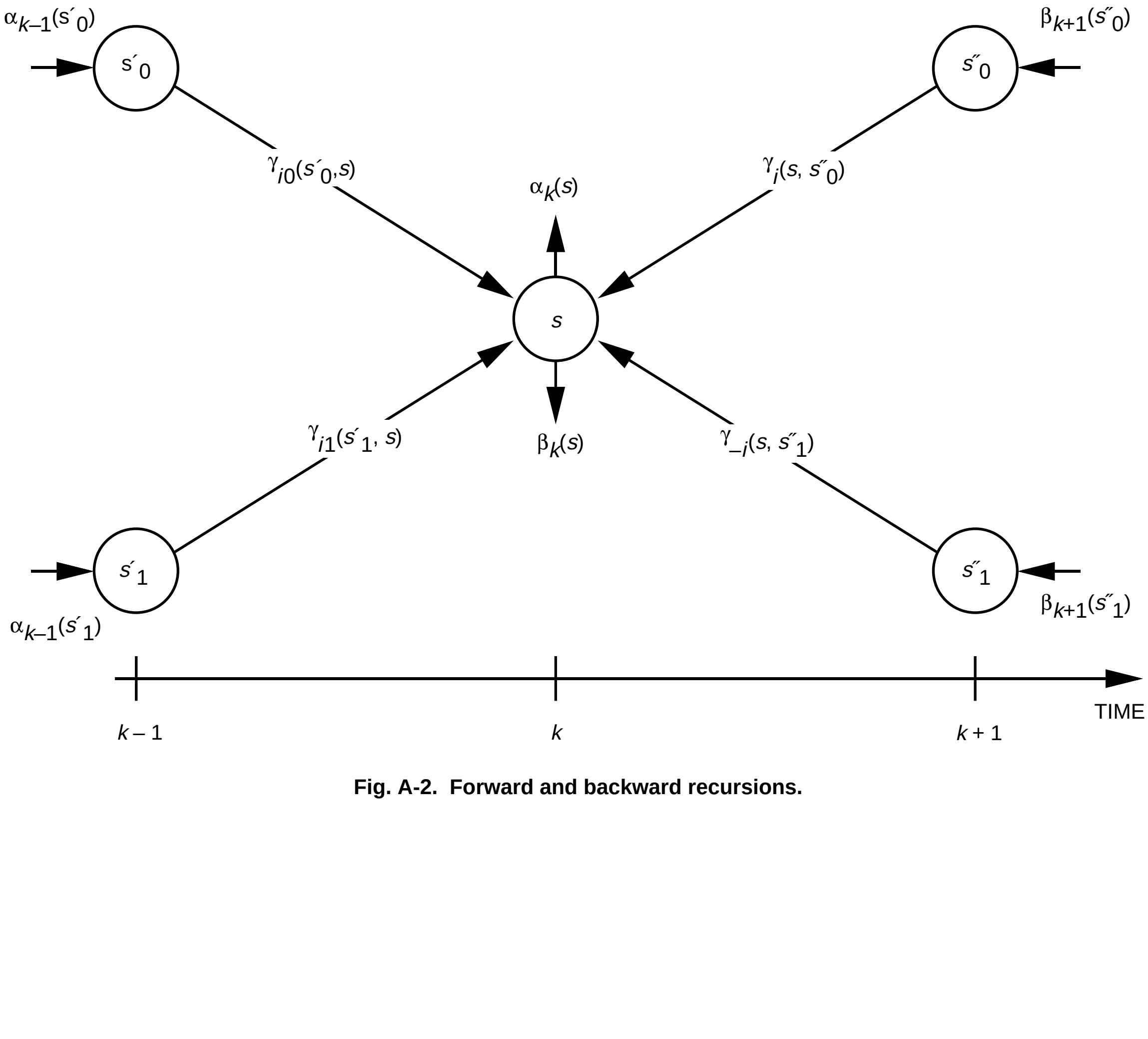

Fig. A-2. Forward and backward recursions.