Code Performance as a Function of Block Size

S. Dolinar,[1] D. Divsalar,[1] and F. Pollara[1]

The recent development of turbo codes has stirred new interest in trying to understand how closely practical codes can approach the theoretical limits on code performance. This article reformulates Shannon's sphere-packing bound in a normalized form conducive to analyzing the relative performance of families of turbo codes. Even though this bound is achievable only by a (generally mythical) perfect *spherical code for the continuous-input additive white Gaussian noise (AWGN) channel, it still can serve as a useful performance benchmark against which all codes of a given rate* r *and information block size* k *can be compared. A measure of* imperfectness *relative to this bound is defined and evaluated for a family of turbo codes and several other well-known codes. Simulations show that these turbo codes approach perfectness closely (within 0.7 dB) and almost uniformly over a wide range of code rates and block sizes. The implication is that turbo codes are near-optimal codes even for block sizes much smaller than previously believed. Finally, the sphere-packing lower bound also is compared to the average performance of randomly selected codes for the continuous-input AWGN channel and to an upper bound on the average performance of randomly selected codes for the binary-input AWGN channel.*

I. Introduction

A. Capacity Limits

The excitement about turbo codes [1] was sparked by their close approach to the ultimate performance limits dictated by channel capacity. Many comparisons have been made between simulated turbo code performance and the ultimate capacity limits for the additive white Gaussian noise (AWGN) channel (both binary-input and unconstrained, continuous-input channels). Table 1 gives these comparisons for a particular family of long-block turbo codes of rates 1/2, 1/3, 1/4, and 1/6 and block length 10,200 information bits that were specified in a preliminary recommendation to the Consultative Committee for Space Data Systems (CCSDS).[2]

The capacity limits in this table show the lowest possible bit signal-to-noise ratio (SNR), E_b/N_0, required to achieve arbitrarily small error probability over the AWGN channel using codes of these rates. A comparison of these limits shows the improvement that theoretically is possible as a result of lowering the code rate. For example, for a binary-input AWGN channel, rate-1/2 codes suffer an inherent 0.7-dB

[1] Communications Systems and Research Section.

[2] A draft of this proposal was presented at the CCSDS coding subpanel meeting, Paris, France, April 14–16, 1997. In a later CCSDS meeting held in South Africa, January 15–17, 1998, the recommended block lengths were specified as 1784, 3568, 7136, 8920, and 16,384 bits.

disadvantage relative to rate-1/3 codes, a 1.0-dB disadvantage relative to rate-1/4 codes, and a 1.8-dB disadvantage relative to the ultimate capacity limit of −1.59 dB for rate → 0. These capacity limits can be adjusted upward to allow for a fixed nonzero allowable bit-error rate [11], but the adjustments are tiny for error-rate requirements below about 10^{-3}.

Just as a constraint on code rate r raises the minimum threshold for reliable communication above the ultimate capacity limit, as listed in Table 1, so does a constraint on a code's information block size k. The theoretical limits shown in Table 1 assume no constraint on block size. Approaching these limits requires that block sizes grow arbitrarily large.

Table 1. E_b/N_0 required for turbo codes defined on 10,200-bit information blocks to achieve a bit-error rate (BER) of 10^{-6}, compared with capacity limits.

Rate	Capacity limit, dB		Turbo code simulation results for BER = 10^{-6}, dB
	Continuous input	Binary input	
1/2	0.00	0.19	0.98
1/3	−0.55	−0.50	0.37
1/4	−0.82	−0.79	0.13
1/6	−1.08	−1.07	−0.12
0	−1.59	−1.59	—

B. Problem Formulation

A classic lower bound on the error probability for codes of a specific block size is the sphere-packing bound developed by Shannon [2]. In this article, we compare the behavior of this bound to the simulated error probabilities achieved by a family of turbo codes of different block sizes.[3] We also compare the sphere-packing bound with the error probabilities achieved by an "average" code of the same block size and rate and by various specific codes thought to be good before the advent of turbo codes.

The problem posed by Shannon assumes a (spherical) block code with $M = 2^k = 2^{rn}$ equal-energy codewords. Each codeword consists of n code symbols with signal energy nE_s per codeword. The channel adds Gaussian noise with variance $N_0/2$ independently to each channel symbol (corresponding to an AWGN channel with two-sided noise spectral density $N_0/2$). In Shannon's formulation of the problem, the "code" constitutes the entire interface between the information and the channel, as shown in Fig. 1(a). For such a code, the output symbols of the code (code symbols) are synonymous with the input symbols to the channel (channel symbols).

In the parlance of modern communication systems, Shannon's code envelops both an error-correcting code and a suitable modulation scheme, as shown in Fig. 1(b). In Fig. 1(b) the code rate, defined as the ratio of the number of information symbols to the number of code symbols, is distinct from the signaling rate, defined as the ratio of the number of information symbols to the number of channel symbols. This distinction disappears in Shannon's model in Fig. 1(a), and we can refer to the code rate or signaling rate interchangeably.

When the information symbols and the code symbols are drawn from same-size alphabets, as is typical for the codes in Fig. 1(b), the code rate r satisfies $0 < r \leq 1$, and $1-r$ measures the fraction of redundancy

[3] A similar evaluation of the sphere-packing bound in [6] does not explicitly illustrate the relationship between the bound and simulated performance of such turbo code families.

added by the code. For Shannon's more general codes, the code-symbol alphabet can be arbitrarily large, and it is convenient to fix the size of the information-symbol alphabet without matching it to that of the code symbols. Thus, in our notation, the amount of information is measured in bits as $k = \log_2 M$, and the code rate, or signaling rate, $r = k/n$, is the number of information bits per code symbol or channel symbol, where a code symbol or channel symbol can be drawn from a continuous alphabet. In these units, code (or signaling) rates can occupy the range $0 < r < \infty$. If restricted to binary codes and a binary-input channel (i.e., both code symbols and channel symbols are binary), there are at most 2^n possible distinct codewords, and in this case code rates $r > 1$ are useless because some of the codewords would have to be repeated.

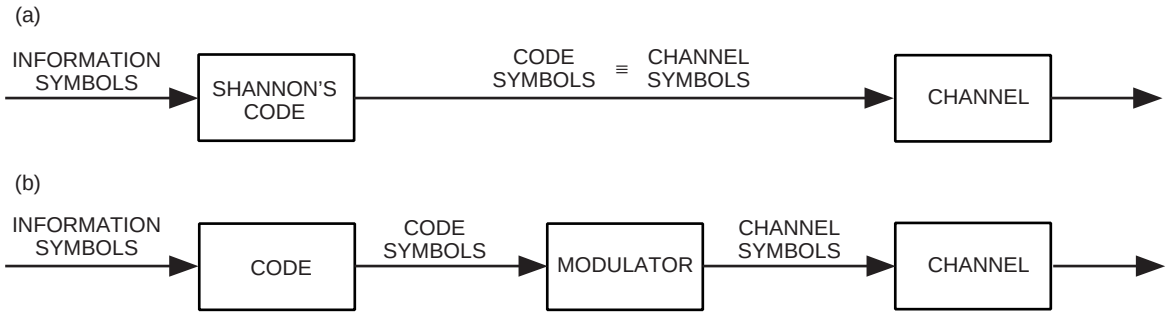

Fig. 1. The problem considered by Shannon: (a) Shannon's model and (b) a typical communication system model.

II. Sphere-Packing Lower Bound on Code Performance

A. The Bound as Computed by Shannon

Shannon [2] derives the sphere-packing bound in the form[4]

$$P_w \geq Q_n(\theta_s, A) \tag{1}$$

where P_w is the codeword-error probability, $A = \sqrt{2E_s/N_0} = \sqrt{2rE_b/N_0}$ is an amplitude corresponding to the channel symbol SNR, $Q_n(\theta, A)$ is the probability that an n-dimensional Gaussian random vector with mean $(A, 0, \cdots, 0)$ and covariance $I_{n\times n}$ (where $I_{n\times n}$ is the $n \times n$ identity matrix) falls outside an n-dimensional cone of half-angle θ around its mean, and θ_s is the angle such that the n-dimensional cone of half-angle θ_s encompasses a fraction $1/M$ of the total solid angle of n-dimensional Euclidean space. Denoting by $\Omega_n(\theta)$ the fractional solid angle within an n-dimensional cone of half-angle θ, then θ_s is the solution to

$$\Omega_n(\theta_s) = \frac{1}{M} \tag{2}$$

Shannon writes exact expressions for the solid-angle function $\Omega_n(\theta)$ and the probability function $Q_n(\theta, A)$:

$$\Omega_n(\theta) = \int_0^\theta \frac{n-1}{n} \frac{\Gamma\left(\frac{n}{2}+1\right)}{\Gamma\left(\frac{n+1}{2}\right)\sqrt{\pi}} (\sin\phi)^{n-2} d\phi \tag{3}$$

[4] In this section and Section III.A, we follow Shannon's problem formulation and solution closely, but some of our notation is slightly different.

and

$$Q_n(\theta, A) = \int_\theta^\pi \frac{(n-1)(\sin\phi)^{n-2}}{2^{n/2}\sqrt{\pi}\Gamma\left(\frac{n+1}{2}\right)} \int_0^\infty s^{n-1} e^{-(s^2+nA^2-2s\sqrt{n}A\cos\phi)/2} ds d\phi \tag{4}$$

Next, he proceeds to derive asymptotic approximations to these functions that are valid for large n:

$$\Omega_n(\theta) \sim \frac{\Gamma(\frac{n}{2}+1)(\sin\theta)^{n-1}}{n\Gamma(\frac{n+1}{2})\sqrt{\pi}\cos\theta} \sim \frac{(\sin\theta)^{n-1}}{\sqrt{2\pi n}\cos\theta} \tag{5}$$

and

$$Q_n(\theta, A) \sim \frac{[G(\theta, A)\sin\theta\, e^{-(A^2 - AG(\theta,A)\cos\theta)/2}]^n}{\sqrt{n\pi}\sqrt{1+G^2(\theta,A)}\sin\theta[AG(\theta,A)\sin^2\theta - \cos\theta]} \tag{6}$$

where $G(\theta, A) = (1/2)[A\cos\theta + \sqrt{A^2\cos^2\theta + 4}]$. Expressions (5) and (6) correspond to Eqs. (28) and (51) in [2].

For large n, the expression in the numerator of Expression (6) dominates exponentially, so Shannon defined an error exponent function, $E_n(\theta, A) \triangleq -(1/n)\ln Q_n(\theta, A)$. The limiting expression for the error exponent becomes

$$E_\infty(\theta, A) \triangleq \lim_{n\to\infty} E_n(\theta, A) = \frac{1}{2}[A^2 - AG(\theta, A)\cos\theta] - \ln[G(\theta, A)\sin\theta] \tag{7}$$

Similarly, defining a corresponding exponent for the solid-angle function, $R_n(\theta) \triangleq -(1/n)\log_2\Omega_n(\theta)$, we have

$$R_\infty(\theta) \triangleq \lim_{n\to\infty} R_n(\theta) = -\log_2(\sin\theta) \tag{8}$$

In principle, one can solve Expression (1) and Eq. (2), using either the exact formulas of Eqs. (3) and (4) or the asymptotic formulas of Expressions (5) and (6), to obtain bounding relationships among the four primary variables, P_w, M, n, and A. For the present application to turbo codes, we are interested primarily in determining, for a given code rate r and desired error rate P_w, how small can the SNR be as a function of the information block length k?

We used Mathematica to evaluate these bounds exactly for small code blocks (n no more than 100 to 200) and asymptotically for larger blocks. The results are shown in Fig. 2 for codes of rates 1/2, 1/3, 1/4, and 1/6 to achieve codeword-error probability $P_w = 10^{-4}$. The figure shows the lower bound on the minimum required E_b/N_0 as a function of k.[5]

[5] In Fig. 2 and succeeding figures, the bounds always are plotted as continuous curves versus k (or r), though the results only make sense at discrete points such that $M = 2^k$ and $n = k/r$ are both integers.

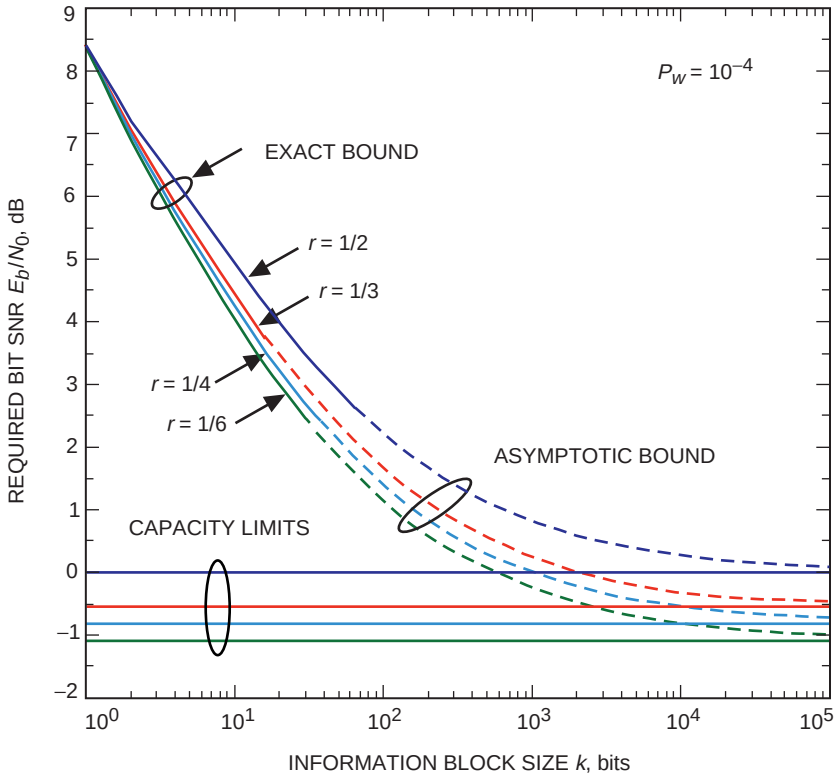

Fig. 2. The minimum required E_b/N_0 implied by the Shannon sphere-packing lower bound for codes with varying information block length k and rates 1/6, 1/4, 1/3, and 1/2, operating over a continuous-input AWGN channel and achieving $P_W = 10^{-4}$.

The solid portion of each curve in Fig. 2 was computed from the exact formulas of Eqs. (3) and (4), while the dashed portion was computed from the asymptotic formulas of Expressions (5) and (6). Also shown are horizontal asymptotes for each code rate, equal to the continuous-input capacity limits in the first column of Table 1. The smooth transitions from the exact computations to the asymptotic computations indicate that the asymptotic formulas already are very accurate (for the purpose of calculating the required SNR) for information block sizes as low as 10 to 100 bits. The ultimate capacity limits are closely approached within 0.1 dB only for block sizes 100,000 and higher.

This figure shows that, for any given code rate, the minimum threshold for reliable communication is significantly higher than the corresponding ultimate limit for that code rate if the code block length is constrained to a given finite size. For example, 1000-bit blocks have an inherent advantage of about 1.4 dB as compared with 100-bit blocks for each of the four code rates plotted. An additional gain of about 0.5 dB potentially is obtained by going from 1000-bit blocks to 10,000-bit blocks, and another 0.2 dB by going to 100,000-bit blocks. After that, there is less than another 0.1 dB of improvement available before the ultimate capacity limit for unlimited block sizes is reached.

B. A Normalized Version of the Bound

Additional insight into the implications of the bound may be obtained by normalizing the required SNR relative to the capacity-limited minimum SNR for arbitrarily long code blocks. Define $A_* \triangleq \sqrt{2^{2r} - 1}$ so that the corresponding $(E_b/N_0)_* \triangleq A_*^2/(2r)$ is the capacity-limited minimum bit SNR for the continuous-input Gaussian channel [10]. These rate-dependent capacity-limited bit SNR values are listed in the first column of Table 1, and they are plotted versus r as the bottommost curve in Fig. 7 later in this article.

Now define ΔSNR_* to be the excess required E_b/N_0 above the corresponding capacity-limited value at the given code rate r, namely $\Delta SNR_* \triangleq (A/A_*)^2 = (E_b/N_0)/(E_b/N_0)_*$. The value of ΔSNR_* represents, for each code rate r, the minimum SNR penalty incurred by constraining the code to have the finite information block size k. Figure 3 gives the same curves as in Fig. 2, but now normalized to show ΔSNR_* rather than the absolute required E_b/N_0 for each code rate. On this graph, the bounds for different code rates are nearly impossible to distinguish for block sizes greater than 10 or 100 bits. By plotting ΔSNR_*, instead of absolute E_b/N_0, versus k, instead of n, we have *virtually eliminated the dependence on code rate* (over the range of code rates plotted, 1/2 to 1/6). The result is a nearly universal curve for predicting the minimum required block length as a function of the excess E_b/N_0 over the (code-rate-dependent) capacity limits in Table 1.

For code rates higher than those plotted in Fig. 3, the bunching of the normalized curves is less dramatic. A quantitative estimate of this effect for large k can be obtained by studying the variation with r of the right side of Expression (11) in the next section. We remark that the normalized curves would stay nearly independent of r for arbitrarily large r if the block size k on the x-axis were replaced by $\min\{k \ln 2, n\}$, i.e., the smaller of the information block size (measured in nats) and the code block size (measured in channel symbols).

C. A Simple Approximation for Large Blocks

In this section, we explore theoretically the reason for the convergence of the normalized bounds in Fig. 3. First make the substitution $A = \gamma \cot\theta$ for the amplitude signal-to-noise ratio in Eq. (4) or Expression (6) for the probability function $Q_n(\theta, A)$. The significance of this substitution stems from the

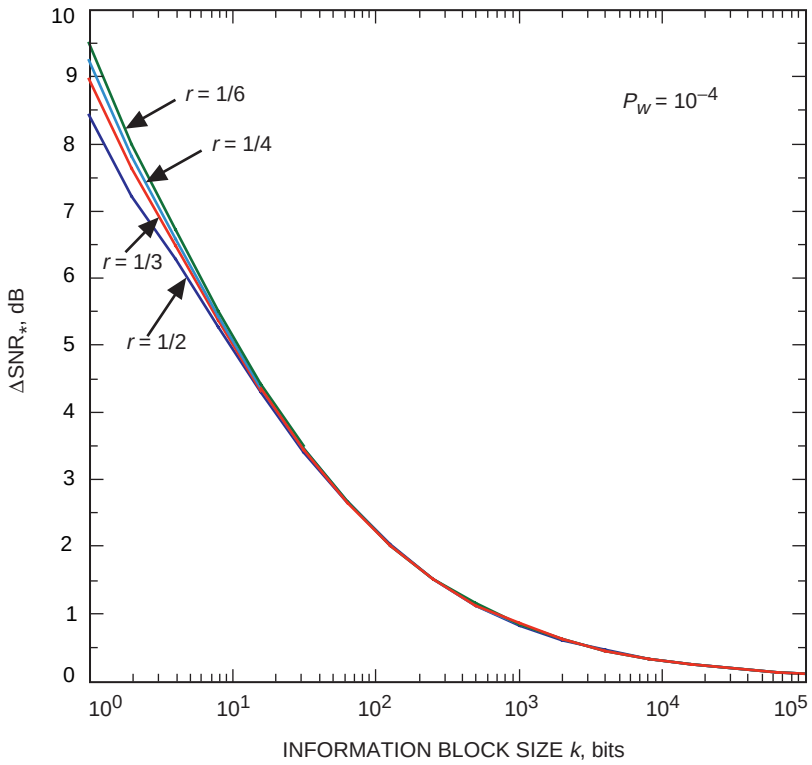

Fig. 3. The minimum required SNR above the code-rate-constrained capacity limit, implied by the Shannon sphere-packing lower bound for codes with varying information block length *k* and rates 1/6, 1/4, 1/3, and 1/2, operating over a continuous-input AWGN channel and achieving $P_W = 10^{-4}$.

fact that $\cot\theta$ is (asymptotically for large n) equal to the capacity-limited value of A for a fixed code rate r, when θ is evaluated at the solution to Eq. (2), i.e., $\lim_{n\to\infty} \cot\theta|_{\theta=\theta_s} = \sqrt{2^{2r}-1} = A_*$. Thus, γ is asymptotically the ratio of the amplitude signal-to-noise ratio A to the corresponding capacity-limited value A_*.

With this substitution, it can be shown that the asymptotic expression of Eq. (7) for the error exponent has the following behavior for γ near 1 ($\ln\gamma$ near 0):

$$E_n(\theta, \gamma\cot\theta) = (\ln\gamma)^2 \frac{1-\sin^2\theta}{1+\sin^2\theta} + O[(\ln\gamma)^3] \tag{9}$$

Keeping only the first term in Eq. (9) and substituting the limiting expression $\sin\theta_s = 2^{-r}$ obtained from Eqs. (2) and (8), we arrive at a particularly simple approximation to the bound, which is valid for large n:

$$\ln P_w^{-1} \leq \left[\ln\left(\frac{A}{A_*}\right)\right]^2 \left(\frac{k}{r}\right)\left(\frac{2^{2r}-1}{2^{2r}+1}\right) \tag{10}$$

Finally, we transform this formula into more convenient decibel units by making the definitions $P_{w,dB}^{-1} \triangleq 10\log_{10}(P_w^{-1})$ and $\Delta SNR_{*,dB} \triangleq 10\log_{10}\Delta SNR_*$, and thus obtain

$$k(\Delta SNR_{*,dB})^2 \geq \left(\frac{40}{\ln 10}\right)\left(r\frac{2^{2r}+1}{2^{2r}-1}\right) P_{w,dB}^{-1} \tag{11}$$

This version of the bound may be approximated further by noting that the function of r in parentheses on the right side of Expression (11) is very insensitive to r for code rates between 0 and 1. This factor increases slowly from $1/\ln 2 = 1.44$ at $r = 0$, to $3/2 = 1.50$ at $r = 1/2$, to $5/3 = 1.67$ at $r = 1$. Thus, for code rates between 0 and 1, the bound may be approximated by a formula independent of r:

$$k(\Delta SNR_{*,dB})^2 \geq (27 \pm 2) P_{w,dB}^{-1} \tag{12}$$

For example, for a required word-error rate of 10^{-4}, this is approximated by a simple handy formula:

$$k(\Delta SNR_{*,dB})^2 \gtrsim 1000 \tag{13}$$

This says that codes with information block lengths on the order of 1000 bits cannot possibly get closer than about 1 dB from the corresponding rate-dependent capacity limits while achieving a word-error rate, P_w, of 10^{-4}. To approach the rate-dependent capacity limits within 0.5 or 0.25 dB at this P_w requires at least 4000- or 16,000-bit blocks, respectively. These required block sizes approximately double, or halve, if the P_w requirement is tightened to 10^{-8} or loosened to 10^{-2}, respectively.

Figure 4 compares the normalized bounds from Fig. 3 with the approximate limiting formulas from Expression (11). We see that the simple formula of Expression (12) or (13) is accurate within 0.2 dB for information block sizes greater than about 1000 bits. For smaller block sizes, this approximation is not very tight, and the exact or asymptotic computations should be used instead.

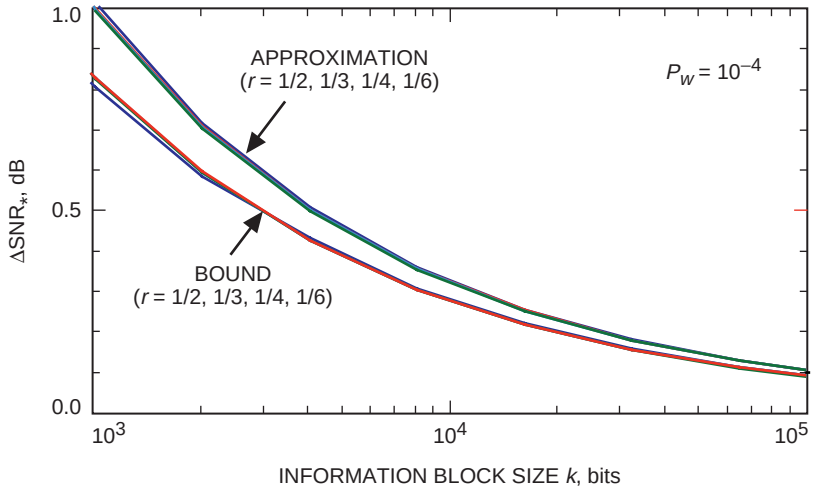

Fig. 4. A comparison of the normalized lower bound with the simple approximation for large blocks.

D. The Bound for Higher Code Rates

Thus far, we have concentrated on evaluating the sphere-packing bound for the range of code rates that historically have been used for deep-space applications, namely $0.15 \leq r \leq 0.5$. In this section, we investigate briefly how the bound behaves at higher code rates.

We extended our evaluation of the sphere-packing bound of Expression (1) to many code rates higher than those plotted in Fig. 2. These extended results are plotted in Fig. 5. This figure shows contour lines of constant required E_b/N_0 to achieve $P_w = 10^{-4}$ for codes with different combinations of code rate r and information block size k. The integer contour values are shown (in red) just above the corresponding contour lines. The contours in this figure were obtained using a sophisticated nonuniform-grid interpolation procedure [9]. Located between the contour lines in this figure are the discrete points at which the bound was explicitly evaluated. These points are depicted as (blue) dots labeled with the corresponding minimum E_b/N_0 value for that particular combination of r and k. The E_b/N_0 values labeling the (blue) dots on the right edge of the figure were computed in the limit as block size k goes to infinity, and they correspond to the capacity-limited values $(E_b/N_0)_*$ as a function of rate.[6]

This figure has a wide range of potential applications. As we shall see in Section V, a graph of E_b/N_0 versus the two fundamental code parameters r and k is a particularly convenient form for use as a code selection tool. For deep-space applications, we are interested primarily in the region for code rates from 0 to 1/2 and for block lengths of 2^{10}=1024 bits and higher. However, the region of higher rates depicted on the y-axis is useful in applications where bandwidth efficiency is paramount. In this case, as mentioned earlier, the signaling rate on the y-axis should be interpreted as the overall spectral efficiency of the combined coding and modulation (measured in bits/s/Hz/dimension).

[6] The required E_b/N_0 in the limit as $k \to \infty$ is the same as the capacity-limited value despite our allowance of a fixed nonzero error rate, $P_w = 10^{-4}$. This contrasts with the results that would be obtained under a fixed nonzero *bit*-error-rate requirement, for which the required E_b/N_0 in the limit as $k \to \infty$ is slightly lower than the corresponding capacity-limited value [11]. The explanation is that a fixed word-error-rate standard is increasingly harder to meet as the word size k increases, and the minimum required E_b/N_0 for achieving $P_w = 10^{-4}$, in the limit as $k \to \infty$, is the same as if we had demanded an arbitrarily small error rate.

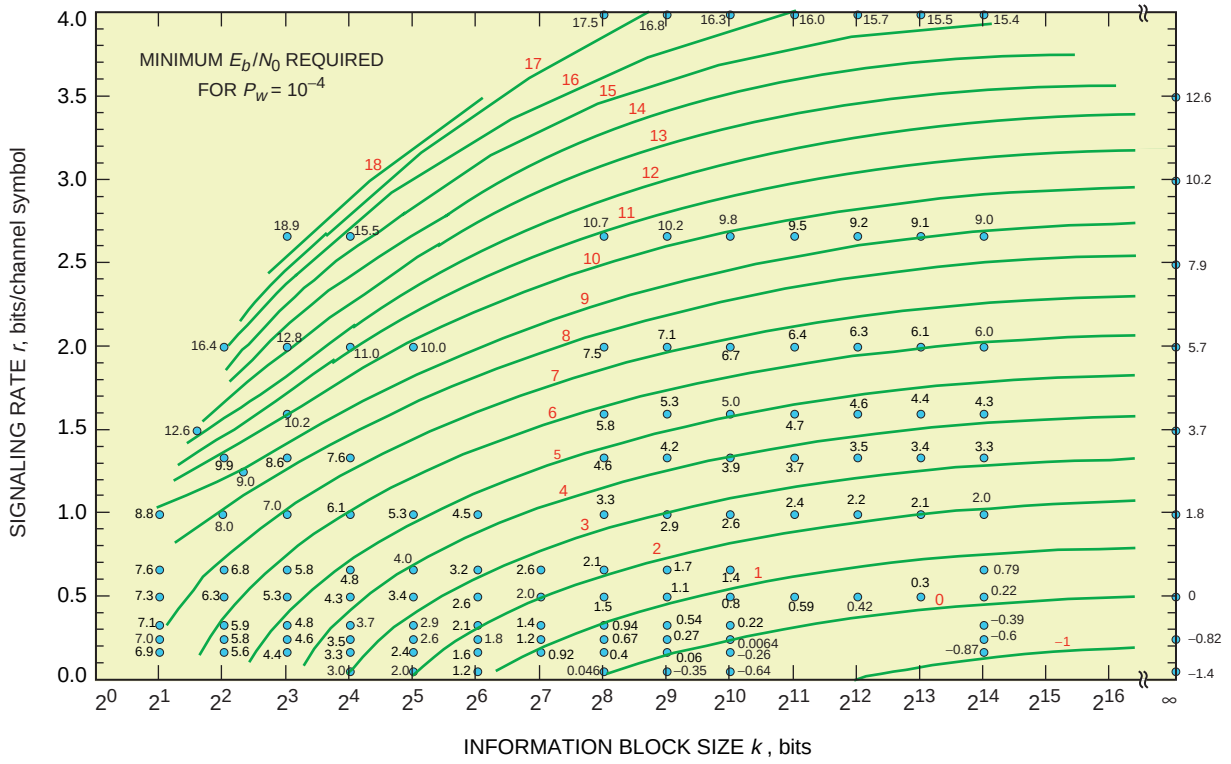

Fig. 5. Contour lines of constant required E_b/N_0 (dB) to achieve $P_w = 10^{-4}$ for different combinations of r and k.

III. Average Code Performance

A. Randomly Selected Codes for the Continuous-Input Channel

Shannon [2] also computed an upper bound on performance by a "random coding" method. His random coding bound gives an exact expression for the *ensemble average* word-error probability, $\bar{P}_w$, averaged over the ensemble of all possible (n, k) spherical codes, when the 2^k codewords are selected independently and completely at random, subject to an energy constraint imposed by the actual signal-to-noise ratio, E_b/N_0. The exact expression for $\bar{P}_w$ can be calculated in terms of the solid-angle function $\Omega_n(\theta)$ and the probability function $Q_n(\theta, A)$ defined in Eqs. (3) and (4):

$$\bar{P}_w = \int_0^{\pi} \left\{1 - [1 - \Omega_n(\theta)]^{M-1}\right\} \left[\frac{-\partial Q_n(\theta, A)}{\partial \theta}\right] d\theta \tag{14}$$

In this equation, $[-\partial Q_n(\theta, A)/\partial\theta]$ is the probability *density* function corresponding to the probability *distribution* function $\{1 - Q_n(\theta, A)\}$.

Shannon derives an asymptotic formula for $\bar{P}_w$ that can be expressed conveniently as the previously derived asymptotic formula of Expression (6) for the lower bound $Q_n(\theta_s, A)$ multiplied by a factor essentially independent of n,

$$\bar{P}_w \sim Q_n(\theta_s, A)\left(1 + \frac{A\,G(\theta_s, A)\sin^2(\theta_s) - \cos(\theta_s)}{2\cos(\theta_s) - A\,G(\theta_s, A)\sin^2(\theta_s)}\right) \tag{15}$$

This expression is valid as long as the angle θ_s, the solution to Eq. (2), is such that both the numerator and the denominator of the fraction in parentheses on the right side of Expression (15) are positive. Inside this range, the multiplicative factor varies from 1 for $\theta_s = \cot^{-1} A_*$ (corresponding to the capacity limit) to ∞ for $\theta_s = \theta_c$ (the value for which the denominator in Expression (15) vanishes). For larger values of θ_s ($\theta_s > \theta_c$), there is an alternative asymptotic expression (see the derivation leading to Eq. (61) in [2]), but all of our evaluations for $P_w = 10^{-4}$ were done using either Eq. (14) or Expression (15).

Figure 6 shows an evaluation of the ensemble average word-error probability, $\bar{P}_w$, using the exact expression of Eq. (14) for information block sizes less than about 50 bits and the asymptotic Expression (15) for larger block sizes for the same code rates plotted in Fig. 2. We verified that the random coding bound is virtually indistinguishable from the sphere-packing bound for information block sizes larger than a few hundred bits.

We also extended our computations as shown in Fig. 6 to additional code rates in the range $0 \leq r \leq 1$. Figure 7 gives another comparison of the random coding bound with the sphere-packing lower bound, now plotted versus code rate for various values of block size k. Again we see the convergence of the two bounds for block sizes of a few hundred bits and higher over the entire range of code rates from 0 to 1. For block sizes under 100 bits, however, the difference is small for $r \to 0$ but widens considerably as $r \to 1$.

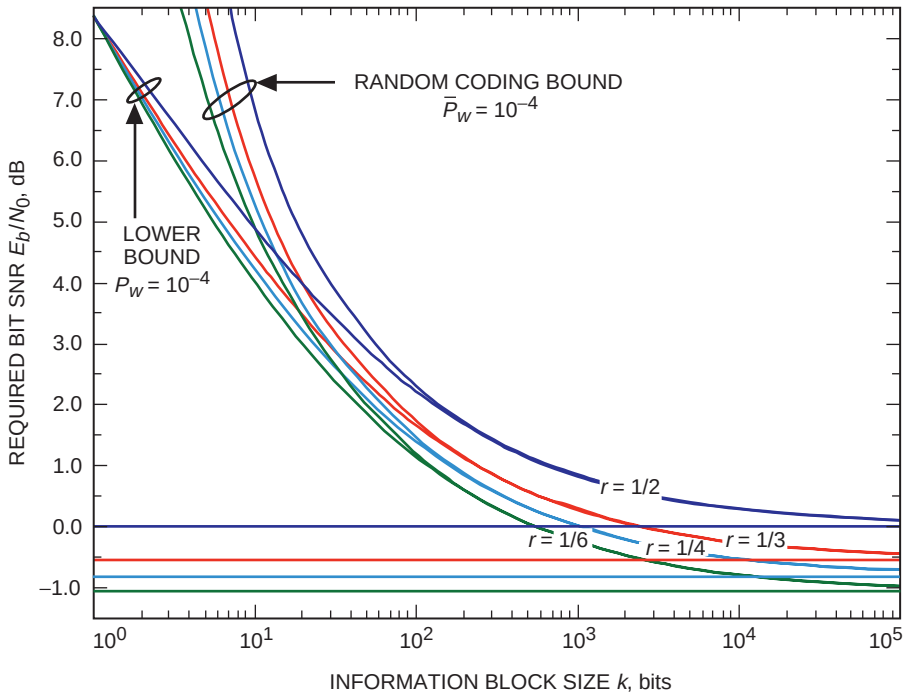

Fig. 6. A comparison of the random coding bound (ensemble average code performance) with the sphere-packing lower bound, as a function of information block size, for various code rates.

B. Randomly Selected Codes for the Binary-Input Channel

The bounds in Sections II and III.A are derived for the AWGN channel with equal-energy codewords following Shannon's formulation of this problem and allowing general spherical codes, as in Fig. 1(a), with code symbols taken from a continuous alphabet. For many applications, it is instructive to study how these bounds change if the codes are constrained to a binary alphabet or, equivalently, if the channel in

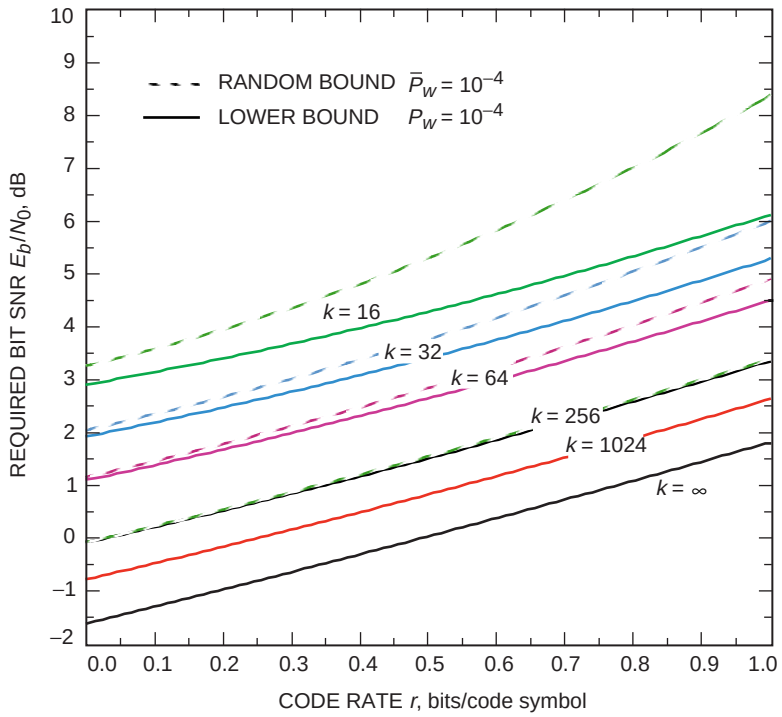

Fig. 7. A comparison of the random coding bound (ensemble average code performance) with the sphere-packing lower bound, as a function of code rate, for various information block sizes.

Fig. 1(a) is constrained to have binary input. When restricted to binary codes and a binary-input channel, code rates $r > 1$ are useless because some of the codewords would have to be repeated.

We set out to analyze how the average word-error probability, $\bar{P}_w$, changes under the constraint of binary codes for a binary-input AWGN channel. Unfortunately, there is no exact expression analogous to Eq. (14) for this constrained $\bar{P}_w$, so we instead evaluated a classic upper bound on $\bar{P}_w$ due to Gallager [3]:

$$\bar{P}_w \le e^{-kE_b/N_0} \left\{ \min_{0 \le \rho \le 1} 2^{\rho r + 1} \int_0^\infty \frac{e^{-y^2}}{\sqrt{2\pi}} \cosh^{1+\rho} \left[\frac{y\sqrt{2rE_b/N_0}}{1+\rho} \right] dy \right\}^n \tag{16}$$

Gallager's bound, Expression (16), is an upper bound on the ensemble average performance of randomly selected (n, k) binary block codes on the binary-input AWGN channel.[7]

The minimum E_b/N_0 satisfying Expression (16) for $\bar{P}_w = 10^{-4}$ is plotted in Fig. 8 versus block size k for the same code rates r as shown in Fig. 6 for the continuous-input channel. Figure 9 shows the same results plotted versus code rate r for various values of block size k (analogously to Fig. 7 for the continuous-input channel).

As expected, the curves in Fig. 8 drop to asymptotic limits that are slightly higher than the corresponding limits in Fig. 6. The asymptotes in Fig. 8 are the well-known capacity limits [11] for the binary-input AWGN channel, and they differ most noticeably from the corresponding asymptotes for the continuous-input channel in the case of $r = 1/2$ among the four rates plotted. Figure 9 focuses on the

[7] Shannon, Gallager, and Berlekamp [4] also proposed a sphere-packing lower bound for discrete memoryless channels. We plan to investigate the application of this bound to the binary-input AWGN channel either by suitable quantization or by evaluating the asymptotic term $o_2(n)$ in [4], which cannot be ignored for moderate and short block sizes.

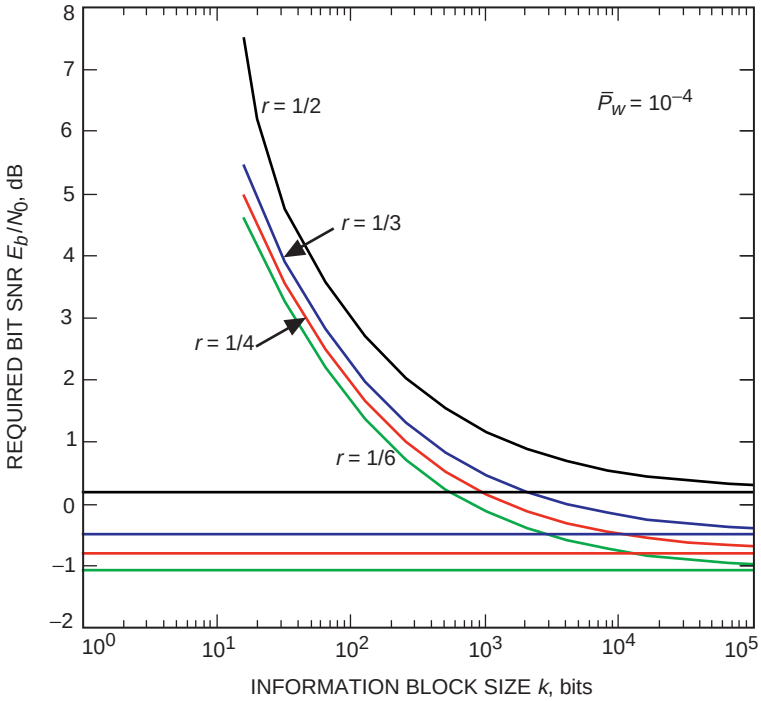

Fig. 8. Gallager's random coding bound (upper bound on ensemble average code performance) for the binary-input AWGN channel, as a function of information block size, for various code rates.

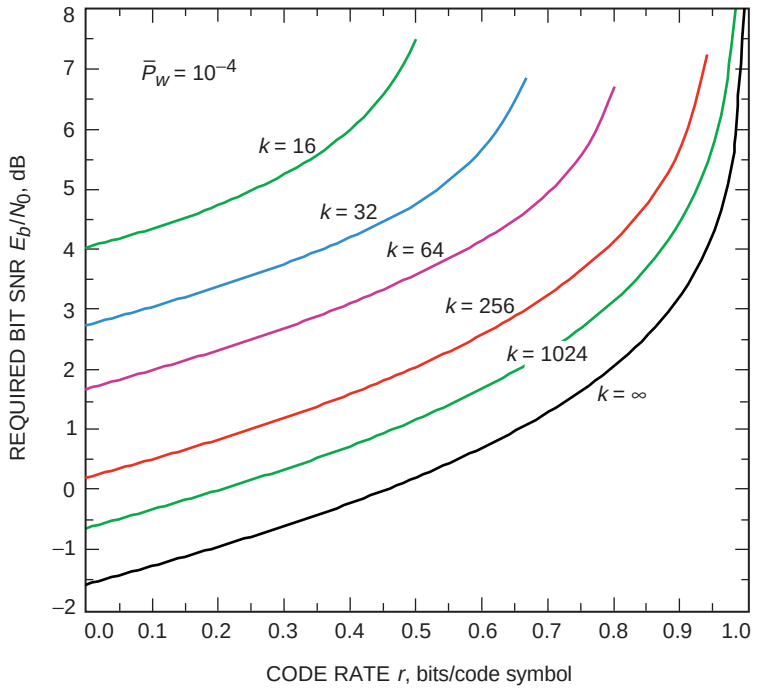

Fig. 9. Gallager's random coding bound (upper bound on ensemble average code performance) for the binary-input AWGN channel, as a function of code rate, for various information block sizes.

rate dependence of the bound in Expression (16), and it is apparent that the required E_b/N_0 increases much more steeply beyond $r = 1/2$ for the binary-input channel than it does for the continuous-input channel in Fig. 7. A quantitative assessment of this difference can be obtained by subtracting the required E_b/N_0 (in dB) shown in Fig. 7 for the continuous-input channel from the required E_b/N_0 (in dB) shown in Fig. 9 for the binary-input channel. This difference is plotted in Fig. 10.

The bottom curve in Fig. 10, for $k = \infty$, is the well-known difference in the capacity limits for the binary-input versus the continuous-input channel. The binary-input constraint costs about 0.2 dB at $r = 1/2$ and negligible amounts at lower rates. For higher rates, the penalty grows rapidly, approaching ∞ as $r \to 1$.

The difference curves for finite block sizes exhibit similar behavior, but the deviations are higher than at the capacity limit. However, we should not attach too much significance to the exact numerical values plotted in Fig. 10 for finite k, because Eq. (14) plotted in Fig. 7 gives the true ensemble average code performance for the continuous-input channel, whereas Expression (16) plotted in Fig. 9 gives only an *upper bound* on the ensemble average code performance for the binary-input channel. Thus, the difference curves in Fig. 10 overestimate the true penalty (averaged over the ensembles of randomly selected codes) for invoking the binary-input constraint, and this overestimate goes to zero only in the capacity limit (the curve for $k = \infty$).

To get a rough idea of how much of the differences plotted in Fig. 10 is due to the binary-input constraint and how much is due to the looseness of the binary-input bound for finite k, we simulated the average performance of randomly selected binary codes of size (32,16). In this case, the required E_b/N_0 to achieve $\bar{P}_w = 10^{-4}$ is only 6.1 dB, whereas the required E_b/N_0 obtained from Expression (16) and plotted in Fig. 9 for $k = 16$, $r = 1/2$, is 7.5 dB. Thus, 1.4 dB of the total difference of 2.2 dB shown in

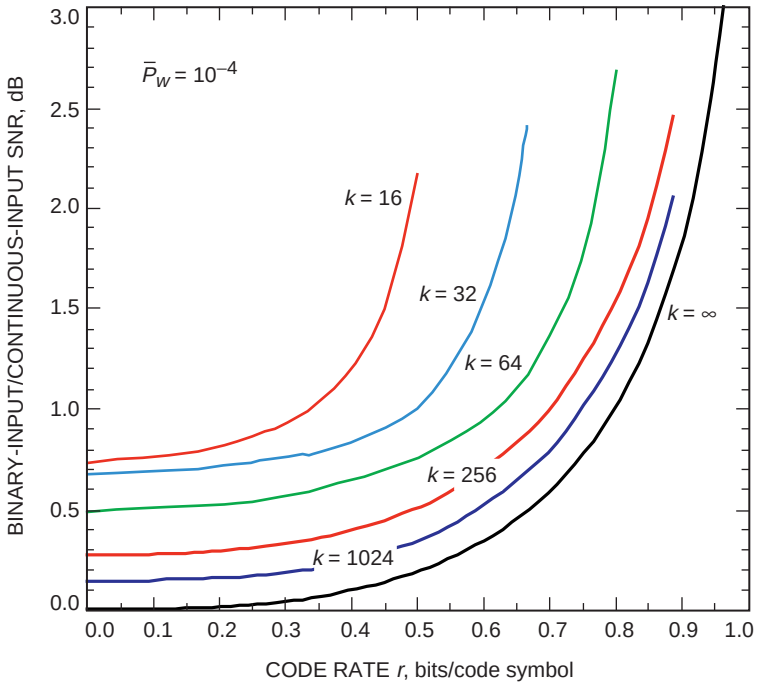

Fig. 10. The difference between Gallager's random coding bound (upper bound on ensemble average code performance) for the binary-input AWGN channel and the random coding bound (exact ensemble average code performance) for the continuous-input AWGN channel.

Fig. 10 for these code parameters results from the looseness of the bound. The remaining 0.8 dB is the penalty caused by requiring a binary-input constraint for the case of $k = 16$, $r = 1/2$, and this penalty is 0.6-dB higher than the corresponding penalty for rate-1/2 codes in the capacity limit.

Another estimate of the looseness of the Gallager bound, Expression (16), for finite values of k can be obtained by observing the y-axis intercepts of the curves in Fig. 10. It can be argued that the required E_b/N_0 to achieve $\bar{P_w} = 10^{-4}$, as rate r goes to zero for any fixed k, should approach the value required to achieve $P_w = 10^{-4}$ using an orthogonal code with 2^k codewords. This result holds for both the continuous-input and the binary-input channels.[8] This implies that all of the curves in Fig. 10 should emanate from the origin if they measured the true SNR difference of the average code performance for the binary-input and continuous-input channels. Thus, the nonzero y-axis intercepts shown in Fig. 10, ranging from about 0.15 dB for $k = 1024$ to 0.75 dB for $k = 16$, are due entirely to the looseness of the Gallager bound.

IV. Performance of Actual Codes

A. Turbo Codes

Simulations by many researchers (see, for example, [5]) have shown that turbo codes can achieve small error probabilities using SNRs just slightly higher than the capacity limits when the turbo code block is very large (10^4 information bits or more). This has prompted many people to jump to the conclusion that turbo codes, therefore, are good only for very large blocks.

It is true that turbo codes defined for smaller block lengths operate farther away from the capacity limit. However, we have seen from Figs. 2 and 3 that the sphere-packing bound mandates a similar relationship between block length and distance from the capacity limit. In this section, we present the results of turbo code simulations demonstrating that these two dependencies mirror each other closely: The required increase in ΔSNR_* predicted by the sphere-packing bound for a given decrease in information block size k is nearly equal to the same variation in ΔSNR_* versus k within a given turbo code "family" (a set of turbo codes of different block sizes but otherwise identical, i.e., based on the same constituent codes).

Figure 11 shows simulation results compared with the lower bounds of Fig. 2 for turbo codes with rates 1/2, 1/3, 1/4, and 1/6 and varying block lengths.[9] In this figure, the turbo codes with information block length 10,200 are the same as those reported in Table 1.[10] For each simulated turbo code, the figure shows the required E_b/N_0 to achieve a codeword-error rate $P_w = 10^{-4}$.

Although there is more than a 2-dB performance differential between the simulation results for the turbo code with $k = 256$, $r = 1/3$, and the code with $k = 10,200$, $r = 1/6$, we see that the difference between the simulations and the corresponding lower bounds remains approximately the same. For block sizes $k = 1024$ and higher, all the simulated turbo codes deviate from their corresponding bounds by about 0.7 dB. For smaller block sizes, this gap widens by a few tenths of a dB to 0.9 dB and 1.05 dB for $k = 512$ and $k = 256$, respectively.

---

[8] For fixed k as r goes to zero, we also have shown that Expression (16) reduces to $\bar{P_w} \leq e^{-k(\sqrt{E_b/N_0} - \sqrt{\ln 2})^2}$ for $\ln 2 \leq E_b/N_0 \leq 4 \ln 2$ and $\bar{P_w} \leq e^{-(k/2)(E_b/N_0 - 2\ln 2)}$ for $4 \ln 2 \leq E_b/N_0$. This upper bound for $r \to 0$ matches an upper bound derived in [10] on the performance of an orthogonal code with 2^k codewords.

[9] We first presented this comparison at the CCSDS Subpanel Meeting, Oberpfaffenhofen, Germany, November 7–8, 1996.

[10] The simulated turbo codes are systematic parallel concatenated codes with two recursive convolutional components. The backward connection vector for both component codes is 10011. The forward connection vector for both component codes and rates 1/2 and 1/3 is 11011. The forward connection vectors for rate 1/4 are 10101 and 11111 (first component code) and 11011 (second component code). The forward connection vectors for rate 1/6 are 11111, 11101, and 10111 (first component code) and 11011 and 11111 (second component code). Puncturing of every other symbol from each component code is necessary for rate 1/2. No puncturing is done for rates 1/3, 1/4, and 1/6.

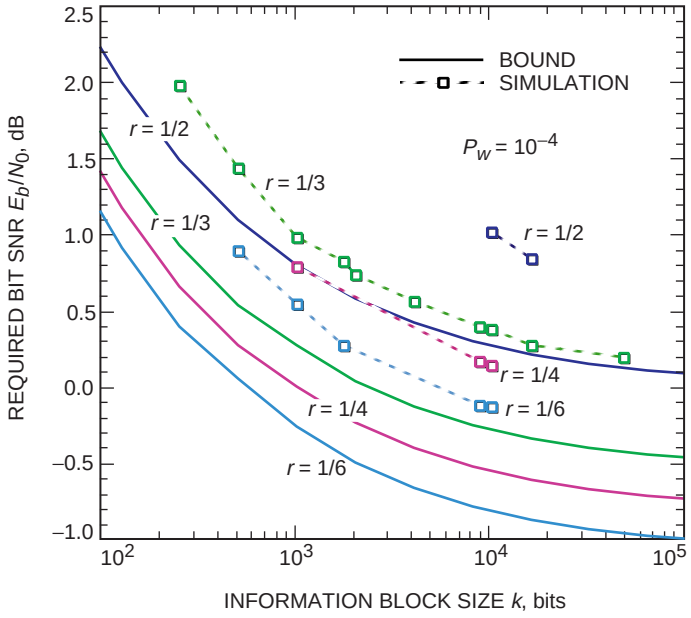

Fig. 11. A comparison of turbo code performance with that of the sphere-packing lower bound, as a function of information block size, for various code rates.

The significance of this half-theoretical, half-empirical result is that we might expect to construct families of turbo codes that approach the ultimate theoretical limits *uniformly* regardless of code rate or block size. For block sizes below about 1000, there might be an underlying theoretical reason that turbo codes will start to progressively deviate from this uniformly close approach to the bound. Alternatively, the uniformly close approach might be maintained by a yet-to-be-discovered tweaking of the turbo code parameters.

In Fig. 6, we saw that the random coding bound nearly coincides with the sphere-packing bound in the range where the performance of the CCSDS family of turbo codes deviates uniformly by 0.7 dB from the sphere-packing bound. This implies that the performance of these turbo codes also deviates by about 0.7 dB from the ensemble average performance of randomly selected codes with the same k and r. Thus, although turbo codes resemble random codes in their construction, this gives a performance measure of the extent to which they are not truly random.

B. Other Good Codes

Figure 12 compares the turbo code performance results from Fig. 11 with the performance of some previously known codes thought to be good before the advent of turbo codes.

The most direct comparisons can be made with the various concatenated coding systems used in deep-space missions for the past two decades. These codes were constructed from an outer (255,223) Reed–Solomon code concatenated with an inner convolutional code, and they employed rectangular interleaving to break up the convolutional decoder's bursty error events. The operations of concatenation and interleaving produce a long block code with information block size k equal to the information block size of the Reed–Solomon code ($223 \times 8 = 1784$ bits) multiplied by the depth I of the interleaving.

In Fig. 12, we show four families of such concatenated codes. Each family is obtained by keeping the component codes constant and varying only the interleaving depth I. The first family is based on the

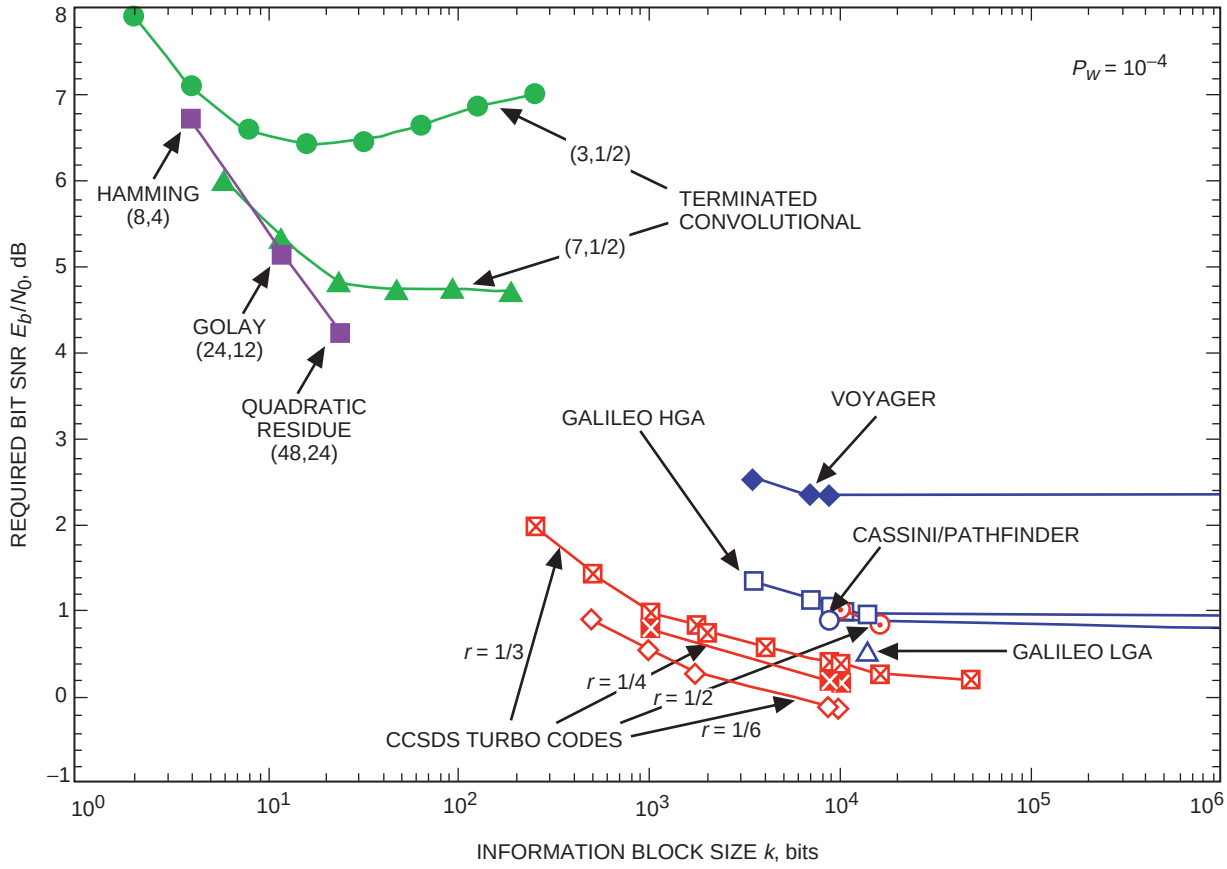

Fig. 12. Simulated performance versus block size for various codes.

CCSDS standard (7,1/2) convolutional code as the inner code; the case $k = 4 \times 1784$ is labeled "Voyager," because the Voyager mission used this code with interleaving depth $I = 4$. The second family is based on the ill-fated (15,1/4) convolutional code that was expected to extend the capabilities of the Galileo mission until the spacecraft's high-gain antenna (HGA) failed to unfurl; the case $k = 2 \times 1784$ is labeled "Galileo HGA." The third family is based on the (15,1/6) convolutional code used for the Cassini and Pathfinder missions; the case $k = 5 \times 1784$ is labeled "Cassini/Pathfinder." For these three families of codes, performance results are plotted for several interleaving depths besides the one actually used, and the performance curves are extended toward $k = \infty$ by using ideal concatenated-coding performance results that assume infinite interleaving ($I = \infty$).

The fourth and final "family" of concatenated codes shown in Fig. 12 consists of one special code [7] designed for Galileo's low-gain antenna (LGA) mission after its high-gain antenna failed to deploy. This code used a (14,1/4) convolutional inner code and a variable-redundancy Reed–Solomon outer code. Unlike the other three concatenated codes, this code was designed to be decoded iteratively in four stages of decoding; the results of the Reed–Solomon decoding in one stage assisted the convolutional decoding in the next stage. There was a fixed interleaving depth, $I = 8$, and the eight outer Reed–Solomon codewords had information block sizes of 161, 245, 225, 245, 195, 245, 225, and 245 8-bit symbols, respectively. The resulting information block size for the entire code block was 14,288 bits.

In Fig. 12, we also have depicted the performance of three other nonconcatenated families of codes often used for very short block lengths. One of these families consists of three famous "best-d_{min}" codes of rate 1/2: the (8,4) extended Hamming code, the (24,12) extended Golay code, and the (48,24) quadratic residue code. These three codes are the smallest known rate-1/2 codes that achieve minimum distance

$d_{min} = 4, 8$, and 12, respectively. The results plotted in Fig. 12 assume maximum-likelihood soft-input decoding for these three codes.

The remaining two nonconcatenated families of codes shown in Fig. 12 are based on plain (nonrecursive) convolutional codes: the CCSDS standard (7,1/2) code with connection vectors $(1111001, 1011011)$ and the (3,1/2) code with connection vectors $(111, 101)$. Normally, convolutional codes are not considered to have a block length, and performance results are reported only as an average bit-error rate. However, for the purpose of comparing convolutional-code performance with the sphere-packing bound, we have reexamined these codes as block codes by terminating each code to various block sizes and evaluating the word-error probability, P_w.

We see by comparing Figs. 12 and 2 that the performance curves for the family of famous best codes and the various families of turbo codes closely parallel the lower bounds for their respective code rates. This is not true, however, for the other families shown. For the concatenated code families, the performance curves quickly flatten out as the block size is increased (by increasing the interleaving depth). Deeper interleaving beyond a certain amount gains insignificantly compared with the available coding gain predicted by the bound for the corresponding increase in block size k. These codes do not fully exploit their large effective block lengths (including interleaving) to achieve the performance gains theoretically possible with increasing k.

For the families of terminated convolutional codes, the required E_b/N_0 to maintain a constant word-error rate bottoms out and then begins to increase slowly as k is increased further. These upward-bending curves depart even farther from the downward-trending bounds than do the flattened curves for the concatenated codes. If a less stringent constant bit-error-rate requirement were substituted, the required E_b/N_0 for convolutional codes also would flatten out as a function of k, but the disparity between this and the bound still is greater, because the convolutional code's performance flattens out in a region of k where the bound is dropping more steeply.

In the next section, we analyze more closely the differences between the actual performance curves in Fig. 12 and the corresponding sphere-packing bounds in Fig. 2 or Expression (1).

C. Evaluation of a Code's Imperfectness

The performance limit corresponding to the sphere-packing bound in Expression (1) would be reached with equality only if the code were a *perfect* spherical code for the continuous-input AWGN channel, i.e., if equal-sized cones could be drawn around every codeword so as to completely fill n-dimensional space without intersecting. Note that perfectness for the continuous-input AWGN problem requires that the entire continuum of n-dimensional Euclidean space be filled by these nonintersecting cones, not just the discrete points represented by binary n-vectors. Thus, under this definition, even the (7,4) Hamming code and the (23,12) Golay code, which are rare examples of perfect binary codes, do not qualify as perfect codes for the continuous-input AWGN channel. Indeed, Shannon [2] mentions that such codes exist only if $k = 1$ or $n = 1$ or 2.

Even though perfectness is an unattainable goal, we already have seen that it can serve as an approachable benchmark for the families of turbo codes considered in Fig. 11. In this section, we define the *imperfectness* of a given code as the difference between the code's required E_b/N_0 to attain a given P_w and the minimum possible E_b/N_0 required to attain the same P_w, as implied by the sphere-packing bound for codes with the same block size k and code rate r. These differences, measured in dB, are shown in Fig. 13 for each of the codes reported on in Fig. 12, with $P_w = 10^{-4}$.

Note that the theoretical reference point for perfectness (measured in dB) consists of two components defined in Section II.B: the rate-dependent, capacity-limited bit SNR, $(E_b/N_0)_*$, and the block-size-dependent (and only slightly rate-dependent) ΔSNR_* plotted in Fig. 3. The imperfectness of any

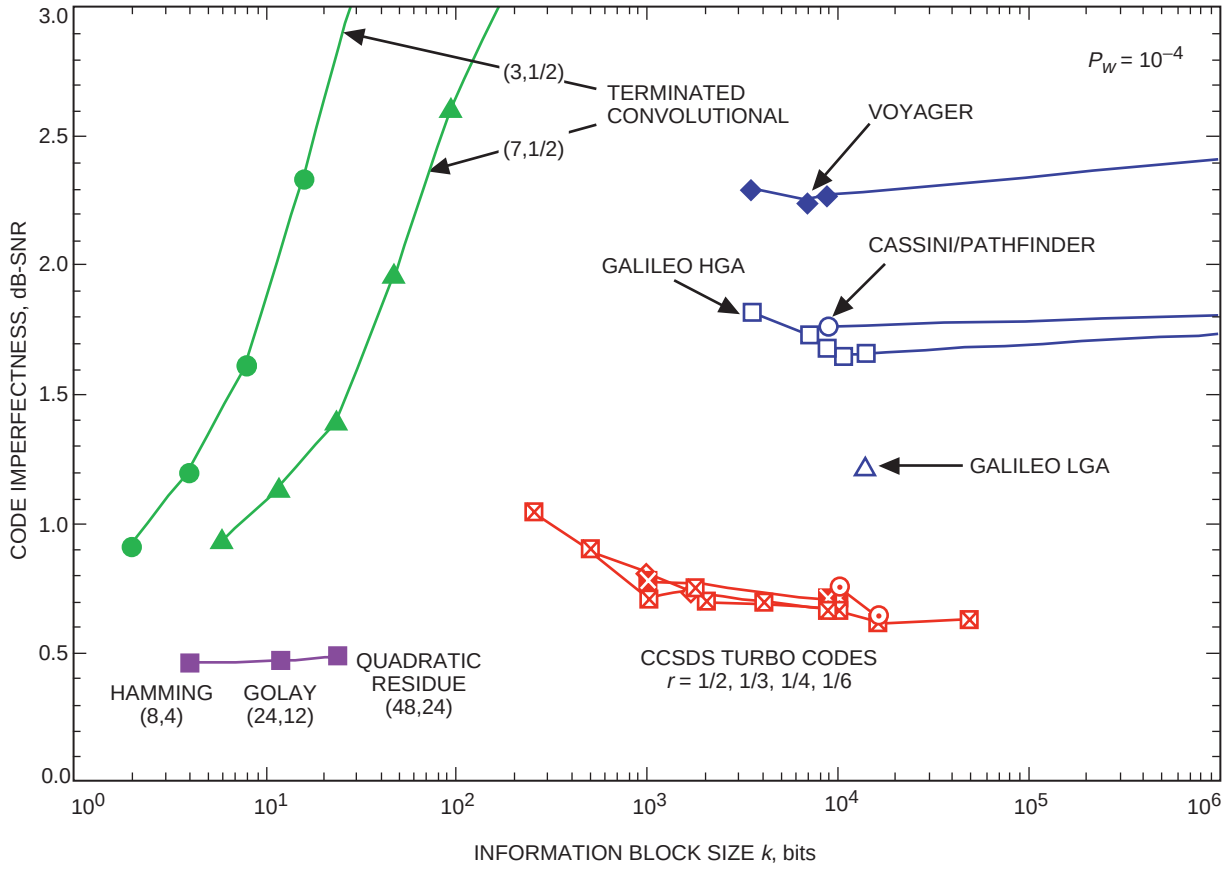

Fig. 13. Code imperfectness relative to the sphere-packing bound, for various codes.

particular code in Fig. 13 is measured in units of dB-SNR, and the amount of imperfectness is the same regardless of whether the SNR refers to bit SNR, symbol SNR, or codeword SNR.

The imperfectness plots in Fig. 13 quantitatively confirm the qualitative conclusions drawn in the preceding section. The three famous best rate-1/2 codes, optimum in terms of minimum Hamming distance for their respective code sizes, are very nearly perfect codes for the continuous-input AWGN channel as well. Each of these codes falls short of perfectness by just under 0.5 dB. As already mentioned in Section IV.A, the turbo code families for rates 1/2, 1/3, 1/4, and 1/6 approach perfectness uniformly by about 0.7 dB for $k > 1000$ and all four code rates.

The curves in Fig. 13 for the two terminated convolutional codes quickly rise away from the perfectness limit as their block sizes increase. A small part of the explanation of this effect is our imposition of a more stringent word-error-rate requirement on codes that were designed to meet an average bit-error-rate requirement. A bigger factor is that convolutional codes are encoded using only a fixed small number of preceding bits and, hence, cannot exploit the potential of large blocks to reduce the required E_b/N_0 at the rate predicted by Fig. 2. These two convolutional code families are most nearly perfect (just under 1 dB of imperfectness) at their smallest possible block sizes, namely, an (8,2) code for the (3,1/2) convolutional family and a (24,6) code for the (7,1/2) convolutional family; not so coincidentally, the terminated convolutional codes with these dimensions also achieve the maximum possible d_{min} among all linear codes of the same size and rate.

We also see from Fig. 13 that JPL's long codes historically have marched toward perfectness in roughly half-dB steps from Voyager to Cassini to the Galileo LGA to future missions that will use turbo codes.

The concatenated code based on the CCSDS standard (7,1/2) code, used by the Voyager mission, misses its perfectness limit by about 2.3 dB. The two concatenated codes based on the very long constraint-length (15,1/6) and (15,1/4) convolutional codes, respectively, used by the Cassini mission and intended for the Galileo HGA mission both miss by about 1.7 dB. For all three of these concatenated codes, the amount of imperfectness varies no more than about 0.1 to 0.2 dB as the interleaving depth is varied. The special four-stage concatenated code used by the Galileo LGA mission [7] approached perfectness within 1.2 dB, and now turbo codes have taken the most recent half-dB step to within 0.7 dB of perfectness (inside the same range of block sizes appropriate for the concatenated codes).

The explanation for this progression is that the constraint-length-15 convolutional codes produced a 0.5 dB to 0.7 dB (depending on interleaving depth) reduction in imperfectness compared with the constraint-length-7 code. However, the price of this performance improvement was a 256-fold increase in decoding complexity. The four-stage Galileo code with its iterative four-stage decoding procedure upped the complexity ante by another factor of two (four stages, but only half as many decoder states per stage) and produced another 0.5 dB reduction in imperfectness. The four-stage Galileo code was in a sense a precursor to two-component turbo codes with their iterative decoding algorithm, and this code was able to move about halfway toward the performance achieved by turbo codes of the same rate and block size. Turbo codes move an additional 0.5 dB toward perfectness because these codes are designed specifically to be iteratively decoded, and their decoders are much more effective at passing soft information throughout all stages of iterations.

Figure 13 also leaves several questions begging for answers or further investigation. If we define (somewhat arbitrarily) codes with imperfectness of less than 1 dB to be "nearly perfect," then we see in Fig. 13 a gap of around an order of magnitude in k, roughly centered around $k = 100$ bits, where no nearly perfect codes are identified. This gap can be filled easily, in principle, by extending the family of famous best-d_{min} codes to larger and larger values of k, using the criterion of maximizing the minimum Hamming distance for each code size. However, the price in decoding complexity would be completely prohibitive. For the three codes in this family shown in Fig. 13, the maximum-likelihood decoding complexities are roughly comparable to those of the three convolutional codes with constraint lengths 3, 7, and 15, respectively. For example, maximum-likelihood decoding of the (48,24) quadratic residue code using a minimal trellis and an optimum permutation of code symbols requires the evaluation of 860,156 trellis edges [8] for every 24-bit block. Thus, it is not practical to construct longer nearly perfect codes by using traditional code design rules. From the other side of the gap, we see that the nearly perfect families of turbo codes start to deviate more substantially from perfectness when their block sizes are less than about $k = 1000$. But these families, unlike the best-d_{min} family, consist of codes with constant decoding complexity per decoded bit. We are optimistic that the range of practically decodable, nearly perfect codes can be extended to smaller values of k by finding turbo codes, or variants such as serially concatenated turbo codes [12,13], that work well for shorter blocks.

The rates of the codes shown in Fig. 13 all fall within the range $0.15 \leq r \leq 0.5$. Outside this range of rates, the quest for nearly perfect codes gets tougher. For $r \to 0$, a code's approach to perfectness is limited practically by a receiver's inability to acquire and detect channel symbols with drastically lowered symbol SNR. For $r > 1/2$, binary codes and modulations no longer suffice, as suggested by the steep climb of the difference between the random coding bounds for the binary-input and continuous-input channels in Fig. 10. For these higher rates, approaching perfectness as defined by the sphere-packing bound for the continuous-input channel will require a combination of good coding and higher-order modulation techniques.

We have seen that, for code rates between 1/2 and 1/6 and block lengths of 1000 bits and higher, we have a family of turbo codes that can approach the limits shown in Fig. 5 uniformly within about 0.7 dB. Outside this region, we have not looked really hard to find good turbo codes, but we still have a smattering of pretty good turbo codes that approach these limits within about 1 or 1.5 dB. Much more work needs to be done in this area. Eventually we might revise our benchmark of perfectness to reflect

additional constraints, such as a constraint imposed by the use of specific modulation schemes, on the possible codes in these other regions.

It also should be pointed out that the imperfectness results in Fig. 13 can change markedly if a different target word-error probability, P_w, is desired. For example, the turbo codes shown here lose their luster of near-perfectness if the word-error-rate requirement is lowered to, say, 10^{-8}, due to the turbo codes' error floor. We are optimistic that nearly perfect codes for error-rate requirements lower than 10^{-4} can be designed using serial concatenation [12–14].

V. Use of the Bounds as a Code Selection Tool

Given the existence of families of turbo codes that approach the performance bounds closely and uniformly over a wide range of code parameters (i.e., block size and code rate), it is reasonable to utilize the theoretical performance bounds as a tool for selecting the actual codes to be used for specific missions. The information block size k and the code rate r are the two most fundamental parameters of the code that affect subsystems external to coding. Bandwidth expansion, channel symbol detectability, and decoding latency are some of the factors that depend only on these two parameters and not on the particular code. Engineers designing external subsystems can now perform fundamental system trade-offs of performance versus block size and code rate merely by reference to a universal table of bounds, because they can be more or less assured of the existence of a practically decodable code that approaches the bound closely at whatever block size and code rate they find desirable to fit their other system constraints.

This approach neatly separates trade-offs that are solely a function of block size and code rate from trade-offs that depend on the particular code. It allows external subsystem engineers to make the most important, most fundamental trade-offs up front, without getting bogged down in understanding the detailed properties of a particular code. These fundamental trade-offs can be calculated simply and accurately, without the need to simulate myriads of codes of different rates and sizes, because they are based on pure theoretical curves.

Such a trade-off analysis based on theoretical bounds is meaningful only if the bounds accurately reflect actual code performance as a function of k and r. As seen in Fig. 13, practically decodable, nearly perfect codes have long been available for tiny blocks ($k \lesssim 24$), and now turbo codes have extended the region of nearly perfect, practically decodable codes to large blocks ($k \gtrsim 1000$). So, at least in these regions, a trade-off analysis of the effects of varying k and r can justifiably be conducted by referring to the theoretical bounds.

The code selection tool most conveniently would be based on a graph such as Fig. 5. The discrete points on this graph, and the continuous contour lines of equal E_b/N_0, correspond to a look-up table of code performance as a function of the two fundamental code parameters k and r. A system engineer simply would refer to this graph to determine how the minimum possible E_b/N_0 varies with different choices of k and r. The overall code selection procedure would consist of the following steps:

(1) Specify a desired codeword-error rate P_w (e.g., $P_w = 10^{-4}$). This selection is dictated by mission requirements.

(2) Before selecting a specific code, decide on reasonable values for two basic code parameters: code rate r and information block size k. Ideally, for best coding performance, $r \to 0$ and $k \to \infty$. However, practical choices for r and k will be constrained by other system considerations (e.g., bandwidth expansion, channel symbol detectability, decoding latency, etc.). To determine the trade-off between these system considerations and coding performance, refer to a theoretical coding limits graph (such as Fig. 5) for codeword-error rate P_w, showing minimum E_b/N_0 as a function of r and k, achievable by a (generally mythical) perfect code $C^*(r, k)$ constrained to have code rate r and block

size k. This graph guides the system designer by showing the relative SNR penalties incurred by choosing different combinations of the two basic code parameters r and k, regardless of the specific code selected.

(3) Select a specific code $C(r, k)$ of rate r and block size k. Additional system considerations include encoder/decoder complexity, decoded error characteristics, and so forth. To determine how coding performance is affected relative to the ideal assumptions of the previous step, refer to a table or graph (such as Fig. 13) listing specific (imperfect) codes $C(r, k)$ and the corresponding amount of imperfectness incurred by using $C(r, k)$ instead of a perfect code $C^*(r, k)$ of the same r and k. This imperfectness represents a relative SNR penalty for using $C(r, k)$ instead of $C^*(r, k)$, and this penalty (in dB) should be added to the minimum E_b/N_0 (in dB) obtained from the theoretical graph in the previous step.

Codes for deep-space applications generally will be selected from the range of code parameters $r \leq 1/2$ and $k \geq 1000$. Figure 14 provides a blow-up of this code parameter region taken from Fig. 5. In this zoomed graph, we see that the contour lines of equal performance are nearly linear and relatively evenly spaced, so the performance trade-offs are particularly simple for the values of r and k most useful for deep-space codes.

Now that perfectness has been shown to be a closely approachable standard over a wide range of code parameters, we anticipate that "coding gain" will fall out of favor as a measurement of the efficacy of a code. Good codes are much closer in performance to a perfect code than to no code at all. Thus, a code's "gain" with respect to uncoded performance is much less meaningful as a selection criterion than the code's "loss" with respect to the ideal performance of a mythical perfect code. Engineers of the future will choose codes based on tables of coding loss or imperfectness rather than coding gain. But, as mentioned at the end of Section IV.C, this future will not fully arrive until the gaps are filled in where the search for nearly perfect codes continues. In particular, finding nearly perfect codes for rates greater than 1/2 will require a combination of good coding and higher-order modulation techniques, and it may be useful to substitute less stringent interim near-perfectness benchmarks that take into account constraints imposed by a particular type of modulation.

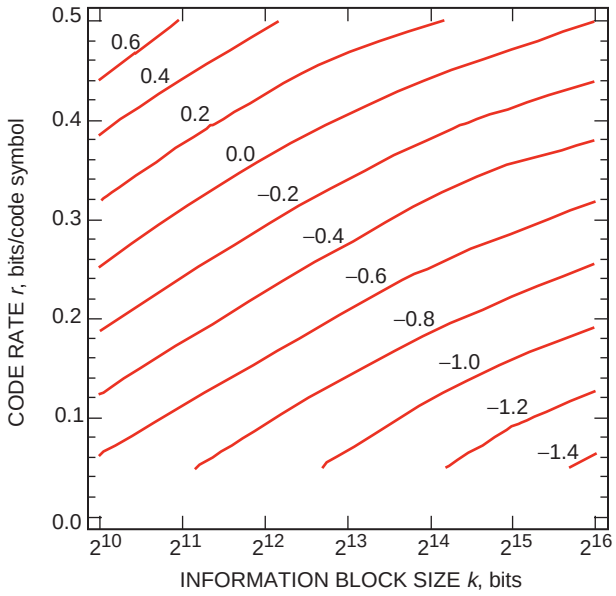

Fig. 14. Contour lines of constant required E_b/N_0 (dB) to achieve $P_W = 10^{-4}$ for different combinations of r and k ($0 \leq r \leq 1/2$ and $2^{10} \leq k \leq 2^{16}$).

Another important caveat mentioned at the end of Section IV.C is that, while the imperfectness graphs succinctly summarize each code's performance, they do not indicate the complexity of decoding each code. All of the codes depicted in Fig. 13 are *practically* decodable, but some are more easily decodable than others. In situations where ease of decoding matters quantitatively, it will be desirable to develop graphs of imperfectness versus decoding complexity to further support a wise code selection.

VI. Conclusion

The bounds evaluated in this article have been well-known for decades. However, the recent development of turbo codes has stirred new interest in trying to understand how closely practical codes can approach the theoretical bounds. The contribution of this article has been to reformulate Shannon's bounds in a form particularly conducive to analyzing the relative performance of families of turbo codes. This reformulation led us to a trivially simple rule of thumb (accurate within 0.2 dB for information block sizes greater than 1000 bits) for estimating how large the information block must be to approach the capacity limit by a given amount. Our simulations showed that a family of similar turbo codes can approach these bounds almost uniformly within 0.7 dB over a wide range of code rates and block sizes. The implication is that turbo codes are very good codes (relative to optimal codes) even for block sizes much smaller than previously believed.

The 0.7-dB degree of imperfectness (relative to the sphere-packing bound) achieved by turbo codes is not unprecedented. For example, we have seen that some classic good codes, designed to maximize the minimum Hamming distance, can approach perfectness even more closely than do these turbo codes. However, we have never (before turbo codes) been able to design practically decodable codes that remain nearly perfect as their block length is increased beyond a few tens of bits. Thus, turbo codes win out over a tiny nearly perfect code, such as the Golay code, not because they are inherently more optimal with respect to codes of their given rate and block length, but because they stay nearly perfect for much longer block lengths, where the bounds allow a drastic reduction in the minimum required E_b/N_0.

In the past, JPL advanced beyond tiny-block codes by building large concatenated codes based on the (255,223) Reed–Solomon code. The information block length for these concatenated codes has ranged from $1784 = 223 \times 8$ bits to $8920 = 5 \times 223 \times 8$ bits, depending on interleaving depth, or even up to 14,288 bits for the variable-length Reed–Solomon concatenated code used for the Galileo LGA mission. These codes gain some of the benefits of large blocks in terms of absolute required E_b/N_0, and, hence, they easily outperform tiny-block codes such as the Golay code. However, they are less perfect with respect to the sphere-packing bound at their respective block lengths, and, thus, they are not able to take maximum advantage of the full theoretical reduction in minimum required E_b/N_0 afforded by their longer blocks. Turbo codes, on the other hand, achieve and maintain a uniform level of near-perfectness over a wide range of large block sizes.

From their inception, turbo codes with block sizes on the order of tens of thousands of bits have taken the coding world by storm because of their close approach to the ultimate capacity limits. Since turbo codes of these sizes perform so outstandingly well, turbo-code advocates sometimes have been seen as trying to convince everyone else to accept a "one size (large) fits all" approach to coding. Now, after recognizing their uniformly close approach to perfectness over a wide range of code rates and block sizes, we see that the true message about turbo codes may be a lot closer to *one code fits all sizes.*

References

[1] C. Berrou, A. Glavieux, and P. Thitimajshima, "Near Shannon Limit Error-Correcting Coding and Decoding: Turbo-Codes," *Proc. of ICC'93*, Geneva, Switzerland, pp. 1064–1070, May 1993.

[2] C. E. Shannon, "Probability of Error for Optimal Codes in a Gaussian Channel," *Bell Syst. Tech. J.*, vol. 38, pp. 611–656, 1959.

[3] R. G. Gallager, *Information Theory and Reliable Communication*, New York: Wiley, 1968.

[4] C. E. Shannon, R. G. Gallager, and E. R. Berlekamp, "Lower Bounds to Error Probability for Coding on Discrete Memoryless Channels," *Info. and Control*, vol. 10, pp. 65–103, 1967.

[5] D. Divsalar and F. Pollara, "On the Design of Turbo Codes," *The Telecommunications and Data Acquisition Progress Report 42-123, July–September 1995*, Jet Propulsion Laboratory, Pasadena, California, pp. 99–121, November 15, 1995. http://tmo.jpl.nasa.gov/tmo/progress_report/42-123/123D.pdf

[6] D. E. Lazic, T. Beth, and M. Calic, "How Close Are Turbo Codes to Optimal Codes," *Intern. Symposium on Turbo Codes*, Brest, France, September 1997.

[7] S. Dolinar and M. Belongie, "Enhanced Decoding for the Galileo Low-Gain Antenna Mission: Viterbi Redecoding With Four Decoding Stages," *The Telecommunications and Data Acquisition Progress Report 42-121, January–March 1995*, Jet Propulsion Laboratory, Pasadena, California, pp. 96–109, May 15, 1995. http://tmo.jpl.nasa.gov/tmo/progress_report/42-121/121Q.pdf

[8] A. Kiely, S. Dolinar, R. McEliece, L. Ekroot, and W. Lin, "Trellis Decoding Complexity of Linear Block Codes," *IEEE Transactions on Information Theory*, vol. 42, no. 6, pp. 1687–1697, November 1996.

[9] T. Wickham-Jones, *Mathematica Graphics*, New York: Springer-Verlag (Telos), 1994.

[10] A. J. Viterbi and J. K. Omura, *Principles of Digital Communication and Coding*, New York: McGraw-Hill, 1979.

[11] S. A. Butman and R. J. McEliece, "The Ultimate Limits of Binary Coding for a Wideband Gaussian Channel," *The Deep Space Network Progress Report 42-22, May–June 1974*, Jet Propulsion Laboratory, Pasadena, California, pp. 78–80, August 15, 1974.

[12] S. Benedetto, D. Divsalar, G. Montorsi, and F. Pollara, "Serial Concatenation of Interleaved Codes: Performance Analysis, Design, and Iterative Decoding," *IEEE Transactions on Information Theory*, vol. 44, no. 3, May 1998.

[13] S. Benedetto, D. Divsalar, G. Montorsi, and F. Pollara, "Analysis, Design, and Iterative Decoding of Double Serially Concatenated Codes With Interleavers," *IEEE Journal on Selected Areas in Communications*, vol. 16, no. 2, pp. 231–244, February 1998.

[14] D. Divsalar, and F. Pollara, "Hybrid Concatenated Codes and Iterative Decoding," *The Telecommunications and Data Acquisition Progress Report 42-130, April–June 1997*, Jet Propulsion Laboratory, Pasadena, California, pp. 1–23, August 15, 1997.
http://tmo.jpl.nasa.gov/tmo/progress_report/42-130/130I.pdf