TURBO CODES FOR NONCOHERENT FH-SS WITH PARTIAL BAND INTERFERENCE [*]

Joseph H. Kang and Wayne E. Stark

University of Michigan

Dept. of Electrical Engineering and Computer Science

Abstract

In this paper, turbo codes are investigated in a slow frequency-hopped spread spectrum (FH-SS) system with partial-band jamming. In addition, full-band thermal noise is present. The channel model is that of a partial-band jammer in which a fraction of the frequency band is jammed and the remaining fraction is unjammed. This paper focuses on the implemention and performance of a modified turbo decoder for this model. We refer to the knowledge that each transmitted bit is jammed as channel state information. We consider cases of known or unknown channel state and variable number of bits per hop. Our approach is to modify the calculation of branch transition probabilities inherent in the original turbo decoder. For the cases with no side information and multiple bits per hop, we iteratively calculate channel state estimates. Analytical bounds are derived and simulation is performed for noncoherent demodulation. The performance of turbo codes is compared with a Reed-Solomon and a concatenated code comprised of a convolutional inner code and Reed-Solomon outer code.

Keywords: Frequency hop communication, Spread spectrum communication, Concatenated coding

[*]This research is supported by the Army Research Office under Grants DAAH04-96-1-0110, DAAH04-96-1-0377, and DAAH04-95-1-0246

1 Introduction

Turbo codes are an exciting new channel coding scheme that achieve data communication at signal-to-noise ratios close to the Shannon limit. The results published in the inaugural paper by Berrou et al [3] were so good that they were met with much skepticism by the coding community. Since then, however, these results have been reproduced and even improved. Consequently, much of the present research is focused on applying turbo codes to different systems.

The turbo encoder is formed using a parallel concatenation of two or more encoders. If the input block has k information bits, the turbo encoded bit stream is made up of the uncoded information bits and the parity bits of the component encoders. The key element of the turbo encoder is the use of an interleaver which permutes the information sequence and then uses this as the input to the second encoder. In general, this permutation allows low weight outputs of the first encoder to result in high weight outputs of the second encoder. Thus, the combination of the encoders might contain favorable distance properties, even if each component encoder does not.

It is well known that a randomly chosen code of sufficiently large block length is capable of approaching channel capacity. In general, however, the complexity of maximum likelihood decoding such a code increases exponentially with block length up to the point where decoding becomes physically unrealizable. The turbo encoder mimics random codes by making use of a large interleaver. While turbo coding performance also improves for increasing interleaver lengths, the decoding complexity grows only linearly, making the decoding of large block lengths possible. Note that the turbo decoder does not perform maximum likelihood decoding directly, but attempts to achieve maximum likelihood decoding in an iterative way. Each constituent decoder uses the MAP algorithm to calculate *a posteriori* likelihood estimates for each bit and then sends this information to the other decoder. This information is thus available to each decoder and is used as *a priori* information to initialize the likelihoods. The turbo decoder iterates through until satisfactory convergence is achieved.

The potential of turbo codes can be best exemplified by its successful application to deep space communications. Turbo codes (16 state constituent codes, overall rate 1/2) were shown to outperform the concatenated code of the Voyager, Galileo, and Cassini missions. The gain, for instance, over the Voyager code (rate 1/2, constraint length 7 convolutional code concatenated with a (255,223) Reed-Solomon code) is approximately 1.5 dB at a BER of 10^{-5}. The primary difference is that the turbo code has greater decoding complexity than the Voyager code.

Thus, we arrive at the primary disadvantage of turbo codes. Decoding complexity for turbo codes is proportional to the block length, the number of decoding iterations, and the constraint length of the constituent codes. Each MAP decoder has the approximate complexity of the Viterbi algorithm and this must be iterated several times. It is known that turbo code performance generally increases with interleaver or block length [8]. In fact, Berrou *et al* showed a bit error rate of 10^{-5} at 0.7 dB, but needed to use 18 decoding iterations and a block length of $65,536$ bits. The large amount of computation required for turbo decoding explains why much of the current research is focused on reduced complexity decoders [10] [11] [12]. By also taking into consideration the processing delay inherent with large interleavers, it is easy to see why turbo codes may be unsuitable for voice communications.

In packet data communications, the use of error correction codes plays a key role in achieving low packet error rates. When transmitting speech, large processing delay is unacceptable and higher error rates are tolerable. In data communication, low error rates are more important and delay at the decoder is more acceptable. Therefore, it would appear that turbo codes are possible candidates for packet data communications.

One packet data network of interest is slow-frequency hop radios. There has been considerable interest in enhancing slow-frequency hop radios so that they can be integrated into a packet radio network [6][14]. One such enhancement would be improved error correction coding. While research on the application of Reed-Solomon codes and concatenated (Reed-Solomon and convolutional) codes to frequency-hopped spread spectrum systems have shown promising results [6][7], it would be interesting to see how well turbo codes perform. Turbo codes were considered for a coherent frequency-hop spread spectrum system with partial band interference in [1]. In this paper, we investigate the performance of turbo codes in a noncoherent frequency-hop spread spectrum system with partial-band interference.

The outline of this paper is as follows. In Section 2, the system model including details of the FH-SS model is described and turbo codes are briefly reviewed. In Section 3, the modifications to the turbo decoder necessary for FH-SS are described. Analytical performance bounds are derived in Section 4. In Section 5, simulation results are presented and compared to the performance of other well-known coding techniques. In Section 6, the numerical results of our analytical bounds are discussed. Finally, we discuss the potential of applying turbo codes to practical FH-SS systems in our conclusion of Section 7.

2 System Model

2.1 Transmitter

As stated before, the turbo encoder is formed using two constituent codes. As in the original work by Berrou *et al* [3], the constituent codes considered in this paper are recursive systematic convolutional codes. The turbo encoder is formed by concatenating the constituent codes in parallel and then separating the codes by an interleaver. A block diagram of the encoder is shown in Figure 1. The encoder input is the data sequence $d_k \epsilon \{0, 1\}$ and the encoder outputs are three streams:

1. The information bits d_k,

2. The parity bits $p_{1,k}$ of the first component encoder with input d_k,

3. The parity bits $p_{2,k}$ of the second component encoder with interleaved d_k as input.

One common problem associated with the implementation of turbo codes is the simultaneous termination of both component encoder trellises. Unlike customary nonrecursive

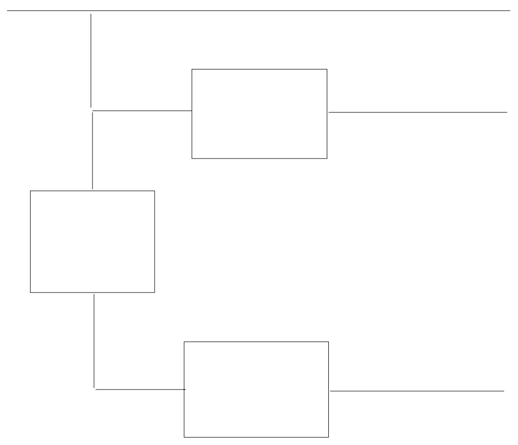
Figure 1: Turbo Encoder

encoders, recursive encoders cannot be driven to the all-zero state by a sequence of M zeros. However, for each of the 2^M possible states, there does exist an M bit code that will drive each encoder to the all-zero state. Unfortunately, at each point in time, the state of each component encoder is generally different due to the existence of the interleaver. To solve this problem, a helical interleaver [13] is considered. In short, a helical interleaver forces both encoders to end in the same state, thus allowing a sequence of M bits to terminate both trellises.

The modulation considered is binary frequency shift keying (BFSK). The resultant signal is frequency hopped. The hopping patterns of the FH-SS system are modeled as sequences of independent random variables uniformly distributed over the allowable frequency range.

2.2 Channel

It is assumed that there exists an on-off jammer that will evenly distribute its power over a fraction ρ of the frequency range. Thus, transmission occurs over a channel that includes both full-band thermal noise with power spectral density $\frac{N_0}{2}$ and partial band interference with power spectral density $\frac{N_J}{2\rho}$ which covers a fraction ρ of the band. As a result, there are essentially two channel states: jammed and unjammed. The probability of hopping to a jammed state is ρ and the probability of hopping to an unjammed state is $1-\rho$. It is assumed that the jammer stays on for the entire duration of the hop if it is jammed at all. Let $(y_{1,k}, y_{2,k}, y_{3,k})$ be the outputs of the channel and let $z_{i,k}$ represent the channel state for $y_{i,k}$. Jammed and unjammed states correspond to $z_{i,k} = 1$ and $z_{i,k} = 0$, respectively.

2.3 Original Turbo Decoder

Turbo decoding is an iterative procedure which makes use of the MAP algorithm. The derivation of this algorithm has been well documented in previous papers [2][3][4]. For notational purposes, the algorithm is briefly reviewed here. The following equations coincide with the first MAP decoder, but can easily be extended for the second MAP decoder. Let S_k be the state of the first encoder at time k, and L_k be the log likelihood ratio (LLR) of the *a posteriori* probabilities. Then

$$\begin{aligned} L_k &= \log \frac{Pr(d_k = 1|\underline{y}_1, \underline{y}_2)}{Pr(d_k = 0|\underline{y}_1, \underline{y}_2)} && (1) \\ &= \log \frac{\sum_m \sum_{m'} \gamma_1(y_{2,k}, m', m)\alpha_{k-1}(m')\beta_k(m)}{\sum_m \sum_{m'} \gamma_0(y_{2,k}, m', m)\alpha_{k-1}(m')\beta_k(m)} && (2) \end{aligned}$$

where the forward and backward recursions can be expressed as

$$\alpha_k(m) = \frac{\sum_{m'} \sum_{i=0}^{1} \gamma_i(y_{1,k}, y_{2,k}, m', m)\alpha_{k-1}(m')}{\sum_m \sum_{m'} \sum_{i=0}^{1} \gamma_i(y_{1,k}, y_{2,k}, m', m)\alpha_{k-1}(m')} \quad (3)$$

$$\beta_k(m) = \frac{\sum_{m'} \sum_{i=0}^{1} \gamma_i(y_{1,k+1}, y_{2,k+1}, m', m)\beta_{k+1}(m')}{\sum_m \sum_{m'} \sum_{i=0}^{1} \gamma_i(y_{1,k+1}, y_{2,k+1}, m', m)\alpha_k(m')} \quad (4)$$

The branch transition probabilities are

$$\begin{aligned} \gamma_i(y_{1,k}, y_{2,k}, m', m) &= p(y_{1,k}|d_k = i, S_k = m, S_{k-1} = m') \cdot \\ &\quad p(y_{2,k}|d_k = i, S_k = m, S_{k-1} = m') \cdot \\ &\quad q(d_k = i|S_k = m, S_{k-1} = m') \cdot \\ &\quad Pr(S_k = m|S_{k-1} = m') && (5) \end{aligned}$$

where $q(d_k = i|S_k = m, S_{k-1} = m') = 1$ if bit i is associated with the given state transition and equals 0 if it is not. $Pr(S_k = m|S_{k-1} = m')$ depends on the *a priori* probabilities of the information bits d_k and are computed as

$$Pr(S_k = m | S_{k-1} = m') = \frac{e^{d_k \cdot L_k}}{1 + e^{L_k}} \tag{6}$$

L_k is termed the *extrinsic* portion of the log likelihood ratio. It is used as *a priori* information in each component of the turbo decoder. The concept of extrinsic information is important in that it prevents information introduced by one of the component decoders to be passed back to that component decoder. If $L^{(2)}$ and $L^{(3)}$ correspond to the extrinsic information generated by the MAP1 and MAP2 decoders, respectively, then the combined MAP LLR for bit d_k is

$$\begin{aligned}\tilde{L}_k &= \log \frac{p(y_{1,k}|d_k = 1)}{p(y_{1,k}|d_k = 0)} + L_k^{(2)} + \\ &\quad \log \frac{\sum_m \sum_{m'} \gamma_1'(y_{3,k}, m', m)\alpha_{k-1}(m')\beta_k(m)}{\sum_m \sum_{m'} \gamma_0'(y_{3,k}, m', m)\alpha_{k-1}(m')\beta_k(m)}\end{aligned} \tag{7}$$

where $\gamma_i'(y_{3,k}, m', m) = \frac{\gamma_i(y_{3,k}, m', m)}{Pr(S_k = m | S_{k-1} = m')}$. The first value is the systematic term; the second is the *a priori* term; and the third is the extrinsic term.

Thus the decoding algorithm is as follows. Each MAP decoder calculates $L_k^{(j)} = \frac{p(d_k=1|\underline{y}_1, \underline{y}_j)}{p(d_k=0|\underline{y}_1, \underline{y}_j)}$. This LLR is then used to generate the *a priori* bit probabilities using (6), which the next decoder uses to calculate a new LLR. This process is iterated through eventually converging to some low bit error rate (BER), where final bit decisions use (7). The structure of the turbo decoder is shown in Figure 2. Note that because the log likelihood ratio is the ratio of *a posteriori* information bit probabilities, the turbo decoding process can be viewed as the iterative improvement of *a posteriori* information bit probabilities. We will use this idea, when we consider the turbo decoder for FH-SS systems.

3 Turbo Decoder for FH-SS Systems

The turbo decoding algorithm is dependent on what information is available to the turbo decoder. We examine the cases where knowledge of the channel state (i.e. jammed or unjammed) is either available or unavailable to the decoder. The case of known channel

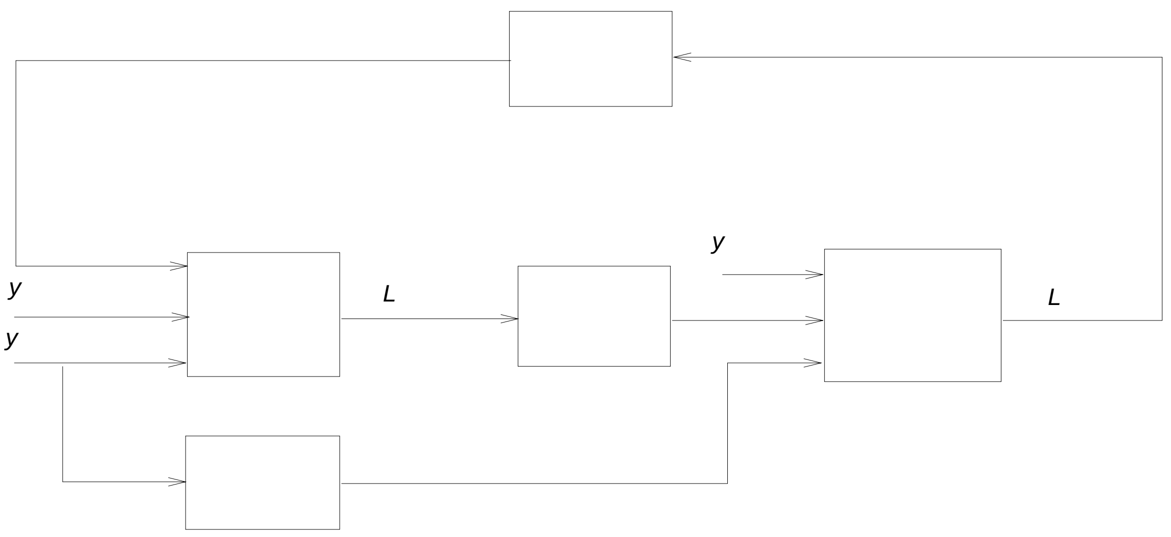

Figure 2: Turbo Decoder

state will be referred to as side information (SI). In addition, the cases of independent and identically distributed (IID) transmission (i.e. one bit per hop) and transmission over a channel with memory (i.e. h bits per hop) are considered. Note that for all cases, channel statistics such as E_b, N_0, N_J, and ρ are assumed to be known to the decoder.

3.1 FH-SS Without Memory

In this section, we consider the case of one bit per hop. If the channel state is unknown, then the modified turbo decoder for IID FH-SS needs only to adapt the calculation of branch transition probabilities. More specifically, (5) is calculated using

$$\begin{aligned} & p(y_{j,k}|d_k = i, S_k = m, S_{k-1} = m') \\ = \quad & p(y_{j,k}|d_k = i, S_k = m, S_{k-1} = m', z_{j,k} = 1) \cdot p(z_{j,k} = 1) + \\ & p(y_{j,k}|d_k = i, S_k = m, S_{k-1} = m', z_{j,k} = 0) \cdot p(z_{j,k} = 0) \qquad (8) \\ = \quad & p(y_{j,k}|d_k = i, S_k = m, S_{k-1} = m', z_{j,k} = 1) \cdot \rho + \\ & p(y_{j,k}|d_k = i, S_k = m, S_{k-1} = m', z_{j,k} = 0) \cdot (1 - \rho) \qquad (9) \end{aligned}$$

If the channel state is known, we treat the side information as information received by the decoder. Thus, for branch transition probability computations, we calculate the joint conditional probability of the channel output, $y_{j,k}$, and the appropriate channel state, $z_{j,k}$.

$$\begin{aligned} & p(y_{j,k}, z_{j,k} = z | d_k = i, S_k = m, S_{k-1} = m') \\ = \quad & p(y_{j,k} | z_{j,k} = z, d_k = i, S_k = m, S_{k-1} = m') \cdot p(z_{j,k} = z) \end{aligned} \tag{10}$$

Having described our modifications for the one bit per hop case, we will move on to the more interesting case with multiple bits per hop.

3.2 FH-SS with Memory

In this section, we discuss our modification to the turbo decoder for the case of multiple bits per hop. First, we detail the design of the hopping structure. The hopping structure is important for cases with memory because unlike the IID case where jammed hops affect single bits, jammed hops now affect multiple bits. For instance, if $c_{1,k}$, $c_{2,k}$, and $c_{3,k}$ are the coded bits which correspond to information bit d_k, then it would be beneficial to send these bits over separate hops. If they were sent over the same hop and that hop was jammed, it would be difficult to decode the information bit correctly. Because the convolutional encoders display memory (i.e. $c_{2,k}$ is dependent on $c_{1,k}$, $c_{1,k-1}$, $c_{1,k-2}$, and so on), it makes sense for similar reasons to separate consecutive coded bits by as much as possible. Using $L = N/h$, the structure of the hopper is shown in Table 1.

For cases with multiple bits per hop and no side information, we attempt to compensate for the lack of side information by generating estimates of the channel. Our approach is to calculate *a posteriori* probabilities for each channel state, $p(z_k | \underline{y}_1, \underline{y}_j)$, and send this information in addition to $p(d_k | \underline{y}_1, \underline{y}_j)$ between decoders. Thus, information bit estimates and channel state estimates can be iteratively improved. The use of channel estimates to calculate branch transition probabilities should lead to improved error rates.

Note that the structure of the hopper shown in Table 1 allows channel estimates to be calculated in a manner similar to the way information bit estimates are calculated in the original turbo decoder. For instance, each MAP decoder in the original turbo decoder calculates *a posteriori* probabilities of the information bits given two of the three received

HOP 1	$c_{1,1}$	$c_{2,L+1}$	$c_{3,2L+1}$	$c_{1,3L+1}$	$\cdots$	$c_{3,N-L+1}$
.	.	.	.	.	...	.
.	.	.	.	.	...	.
HOP L	$c_{1,L}$	$c_{2,2L}$	$c_{3,3L}$	$c_{1,4L}$	$\cdots$	$c_{3,N}$
HOP $L+1$	$c_{2,1}$	$c_{3,L+1}$	$c_{1,2L+1}$	$c_{2,3L+1}$	$\cdots$	$c_{1,N-L+1}$
.	.	.	.	.	...	.
.	.	.	.	.	...	.
HOP $2L$	$c_{2,L}$	$c_{3,2L}$	$c_{1,3L}$	$c_{2,4L}$	$\cdots$	$c_{1,N}$
HOP $2L+1$	$c_{3,1}$	$c_{1,L+1}$	$c_{2,2L+1}$	$c_{3,3L+1}$	$\cdots$	$c_{2,N-L+1}$
.	.	.	.	.	...	.
.	.	.	.	.	...	.
HOP $3L$	$c_{3,L}$	$c_{1,2L}$	$c_{2,3L}$	$c_{3,4L}$	$\cdots$	$c_{2,N}$

Table 1: Structure of Hopper

observation vectors. This information is passed to the next MAP decoder which uses this information as *a priori* information bit probabilities. Similarly, by using the structure of the hopper in Table 1, two-thirds of the relevant observation sequence is available to each MAP decoder so that *a posteriori* probabilities for each hop can be calculated. This information can be passed between MAP decoders and be used as *a priori* channel state probabilities.

If we define $\underline{R}$ as the vector of received channel outputs that is available to the MAP decoder, $\underline{R}_k$ as the subset of $\underline{R}$ that has been received over a given hop with state z_k, and $\tilde{\underline{R}}_k$ as the subset of $\underline{R}$ that has not been received over the hop with state z_k, then calculation of state estimates is as follows.

$$\begin{aligned} p(z_k = z|\underline{R}) &= \frac{p(\underline{R}|z_k = z) \cdot p(z_k = z)}{p(\underline{R})} && (11) \\ &= p(\underline{R}_k|z_k = z) \cdot p(z_k = z) \cdot \frac{p(\tilde{\underline{R}}_k)}{p(\underline{R})} && (12) \\ &= p(\underline{R}_k|z_k = z) \cdot p(z_k = z) \cdot K && (13) \\ &\approx p(\underline{R}_k|z_k = z) \cdot \tilde{p}(z_k = z|\underline{R}_k) \cdot K && (14) \end{aligned}$$

where K is a normalizing factor chosen to make the probability density function sum to 1 and $\tilde{p}(z_k = z|\underline{R}_k)$ is the channel estimate provided by the previous MAP decoder.

In the above equation $\tilde{p}(z_k = z|\underline{R}_k)$ is used in place of $p(z_k = z)$ to take advantage of the state estimate provided by the previous MAP decoder. In order to compute each state estimate, we must calculate the conditional joint probabilities of $\underline{R}_k$ in (14). This can be calculated by performing total probability on $p(\underline{R}_k|z_k = z)$ over the respective coded bits.

$$p(\underline{R}_k|z_k = z) = \sum_{\underline{c}_k} p(\underline{R}_k|\underline{c}_k = \underline{c}, z_k = z)\, p(\underline{c}_k = \underline{c}) \tag{15}$$

where $\underline{c}_k$ represents the vector of coded bits respective to $\underline{R}_k$.

This will require *a priori* probability knowledge of the coded bits. But, the MAP decoders are already sending log likelihood ratios that give $p(d_k = d|\underline{R})$. Using this information, $p(c_{i,k} = c|\underline{R})$ can be calculated. Thus, as the MAP decoders refine their estimates of $p(d_k = 1|\underline{R})$, estimates of $p(z_k|\underline{R})$ are also getting more refined.

Note that as h, the number of bits per hop, increases, the complexity of directly computing $p(\underline{R}_k|z_k = z)$ rises exponentially. To overcome this problem, we developed the following recursion.

1. Base Case

$$\begin{aligned} & p(z_k = z|R_{k,1}) \\ &= \frac{p(R_{k,1}|z_k = z) \cdot p(z_k = z)}{p(R_{k,1})} & (16) \\ &= \sum_{c=0}^{1} p(R_{k,1}|z_k = z, c_{k,1} = c)p(c_{k,1} = c) \cdot \frac{p(z_k = z)}{p(R_{k,1})} & (17) \\ &\approx \sum_{c=0}^{1} p(R_{k,1}|z_k = z, c_{k,1} = c)p(c_{k,1} = c) \cdot \frac{\tilde{p}(z_k = z|\underline{R}_k)}{p(R_{k,1})} & (18) \end{aligned}$$

where $\tilde{p}(z_k = z|\underline{R}_k)$ is the channel state estimate of the previous MAP decoder.

2. Recursion

$$\begin{aligned} & p(z_k = z|R_{k,1}, R_{k,2}, ..., R_{k,i-1}, R_{k,i}) \\ &= \frac{p(R_{k,i}|z_k = z, R_{k,1}, ..., R_{k,i-1}) \cdot p(z_k = z|R_{k,1}, ..., R_{k,i-1})}{p(R_{k,i}|R_{k,1}, ..., R_{k,i-1})} & (19) \\ &= \frac{p(R_{k,i}|z_k = z) \cdot p(z_k = z|R_{k,1}, ..., R_{k,i-1})}{p(R_{k,i}|R_{k,1}, ..., R_{k,i-1})} & (20) \end{aligned}$$

Once state estimates have been computed, they are ready to be used in the turbo decoder. We use state estimates in a manner analogous to the way that the turbo decoder uses information bit estimates. When there is no SI, the *a priori* state probabilities are replaced by the *a posteriori* state probabilities for branch transition probability calculations. As in the IID case, the appropriate *a priori* probability is used for cases with side information. Thus, for the $MAP1$ decoder with $j = 1, 2$,

$$\begin{aligned}
&p(y_{j,k}|d_k = i, S_k = m, S_{k-1} = m') \\
&\quad= p(y_{j,k}|d_k = i, S_k = m, S_{k-1} = m', z_{j,k} = 1) \cdot p(z_{j,k} = 1) + \\
&\quad p(y_{j,k}|d_k = i, S_k = m, S_{k-1} = m', z_{j,k} = 0) \cdot p(z_{j,k} = 0) &(21)\\
&\quad\approx p(y_{j,k}|d_k = i, S_k = m, S_{k-1} = m', z_{j,k} = 1) \cdot p(z_{j,k} = 1|\underline{y}_1, \underline{y}_3)) + \\
&\quad p(y_{j,k}|d_k = i, S_k = m, S_{k-1} = m', z_{j,k} = 0) \cdot p(z_{j,k} = 0|\underline{y}_1, \underline{y}_3)) &(22)
\end{aligned}$$

when there is no side information and

$$\begin{aligned}
&p(y_{j,k}, z_{j,k} = z|d_k = i, S_k = m, S_{k-1} = m') \\
&\quad= p(y_{j,k}|z_{j,k} = z, d_k = i, S_k = m, S_{k-1} = m') \cdot p(z_{j,k} = z) &(23)
\end{aligned}$$

when there is side information.

Note that in the IID (one bit per hop) case, state estimates were not computed. This would be impractical, since the estimate would be based on only one bit of information. With memory, there are more bits of information, thus allowing for a more reliable estimate. One potential disadvantage for the memory case is that an inaccurate state estimate affects multiple bits.

4 Analytical Bounds

It is often impractical to achieve simulated results for extremely low BERs. As a result, bounds are often calculated. Here, we invoke the Union-Bhattacharrya Bound to obtain an upper bound on the probability of error.

Turbo codes are linear, so without loss of generality, we will assume that the all-zeros codeword was transmitted. If A_d is the weight enumerator of the code, $P_2(d)$ is the pairwise error probability between the all-zeros codeword and a codeword of weight d, and D is the Bhattacharrya parameter where $P_2(d) \leq D^d$, then the bound for an (n, k) block code is

$$P_{word} \leq \sum_{d=d_{min}}^{n} A_d \, P_2(d) \tag{24}$$

$$\leq \sum_{d=d_{min}}^{n} A_d \, D^d \tag{25}$$

It was shown in [5] that with noncoherent reception, optimal decoding with side information leads to

$$D = \begin{cases} [\int_0^\infty u e^{-u^2/2} I_0^{1/2}(u\sqrt{2E_s/N_J})du]^2 & E_s/N_J < 2.871 \\ \frac{1.424}{E_s/N_J} & E_s/N_J \geq 2.871 \end{cases} \tag{26}$$

Square-law combining is a suboptimal method of decoding in AWGN, but has been shown to have an approximate performance loss of 0.14 dB for reasonable SNRs [5]. Because square-law combining is suboptimal, an upper bound on its performance will also be an upper bound to the performance of optimal decoding. The Bhattacharrya parameter for square-law combining in worst case jamming is ($\Gamma = E_s/N_J$)

$$D = \begin{cases} [\frac{1}{(1-\lambda)^2} \exp\{-\frac{\lambda\Gamma}{(1+\lambda)}\}] & \Gamma < 3 \\ \frac{4e^{-1}}{\Gamma} & \Gamma \geq 3 \end{cases} \tag{27}$$

$$\lambda = \frac{\sqrt{(2+\Gamma)^2 + 8\Gamma} - (2+\Gamma)}{4} \tag{28}$$

The only known way to exactly calculate A_d is via an exhaustive search involving all possible input sequences. One solution is to calculate an average upper bound by computing an average weight function over all possible interleaving schemes [9]. If the average weight function is defined as

$$\overline{A}_d = \sum_{i=0}^{k} \binom{k}{i} p(d|i) \tag{29}$$

where $p(d|i)$ is the probability that an interleaving scheme maps an input weight of i to produce a codeword of total weight d and $\binom{k}{i}$ is the number of input frames with weight i. An algorithm for calculating $p(d|i)$ was described in [9].

Thus,

$$\overline{P}_{word} \leq \sum_{d=d_{min}}^{n} \sum_{i=1}^{k} \binom{k}{i} p(d|i)\, D^d \tag{30}$$

5 Simulation Results

For all simulations, the component encoders are rate $\frac{1}{2}$ recursive systematic convolutional encoders with memory 4 and octal generators $(37, 21)$. The packet size is 1760 bits and the number of turbo code iterations is 5. The SNR of the full-band thermal noise is set to 20 dB. Cases with memory are simulated using 20, 80, and 160 bits per hop (BPH).

Figure 3 shows the plot of minimum E_b/N_J needed to achieve a packet error rate (PER) of 10^{-3} for a given ρ. As would be expected, the cases with side information (SI) performed better than their counterparts without SI. The SI and no SI curves only meet when $\rho = 1.0$. In this case, all states are jammed, so side information does not provide any additional information.

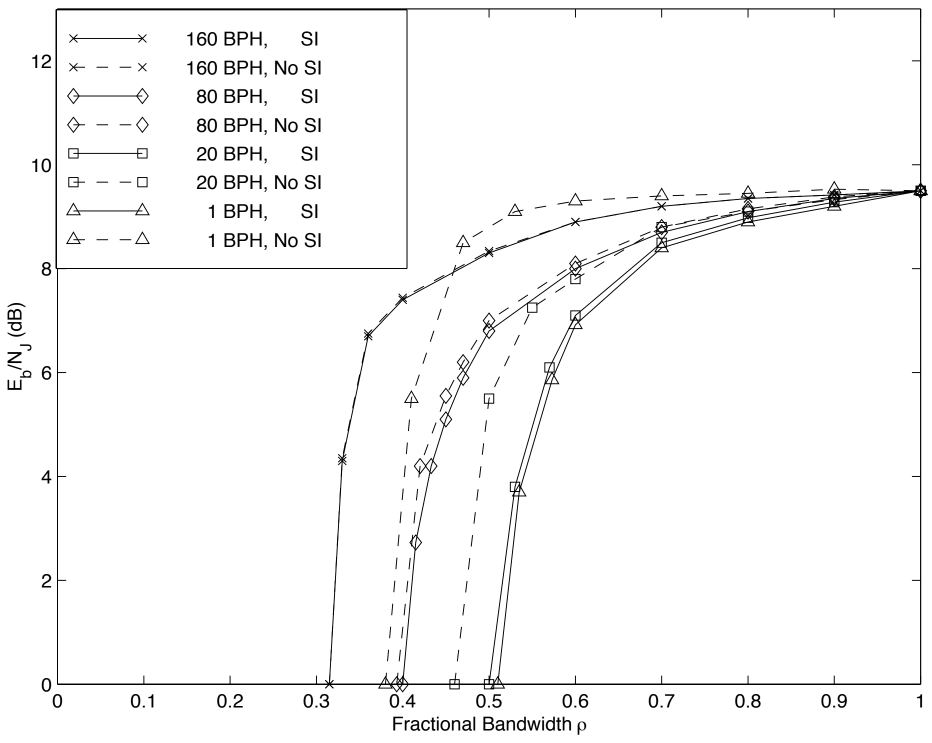

Figure 3: Noncoherent Reception of Turbo Codes

The graphs in Figure 3 show a tradeoff as the number of bits per hop increases. For any memory, no SI case, where channel states are iteratively estimated, performance is obviously upper bounded by the memory SI case. Because channel state estimates will improve if the number of bits per hop increases, we should be able to get arbitrarily close to the corresponding memory, SI result by increasing the memory. This is shown in Figure 3 by examining the performance differences between corresponding SI and no SI cases for variable bits per hop. For memory, no SI cases, we were able to calculate reliable state information. These state estimates provided useful information which in turn aided the decoding process.

The downside of increasing the memory can be seen by analyzing the SI cases in Figure 3. For all SI cases, knowledge of the channel state nulls out the advantage of more effective estimation. As a result, one might expect the performance of SI cases to be similar. However, as shown in Figure 3, this is clearly not the case. The performance of SI cases degrades as the memory increases. Thus, while increased memory leads to improved channel estimates in cases without SI, the upper bound for the performance of no SI cases appears to effectively decrease.

Although SI cases may have equivalent information, direct comparison of cases with different number of bits per hop is unfair. The memory cases have h bits per hop, while the IID case has 1 bit per hop. A fair comparison would require that the total number of hops be equal. In the memory case, the block length was N bits, so there were $N/(h*R)$ hops. Thus, to make the comparison fair, each packet would consist of h sub-codes of N/h information bits each, where each hop contains one coded bit from each of the sub-codes. A packet error would be declared if any of the sub-codes yields an error. Figure 4 shows this system which would yield a more fair comparison.

As shown in Figure 4, increasing the number of bits per hop results in sub-codes of shorter block length. Because turbo codes show improved performance for larger block lengths, it follows that increasing the number of bits per hop results in degradated performance.

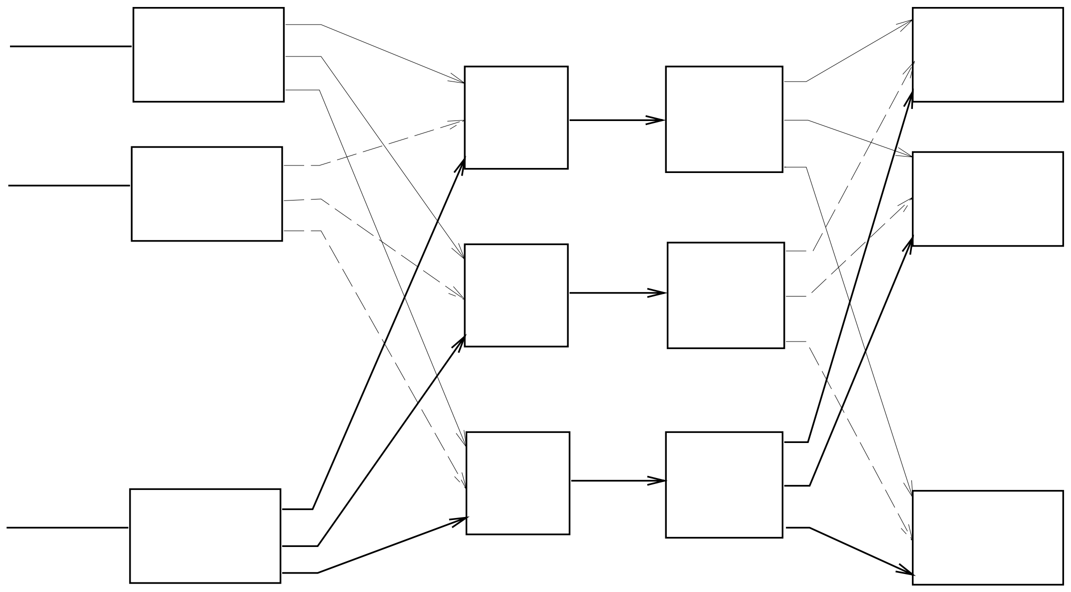

Figure 4: Fair Comparison Between Memory and IID Cases

We have shown that it is possible to bridge the gap between cases with and without side information by iteratively computing state estimates for a large number of bits per hop. However, the performance improvements may still be unsatisfactory. In the case of noncoherent reception, one possible improvement would be to perform phase estimation. If the phase during a hop is assumed to move very slowly and an orthogonal signal set is used, phase estimation can be performed using a method analogous to state estimation. In this manner, joint decoding and phase tracking can be achieved.

In order to gauge our results, we refer to the application of other coding methods to FH-SS systems. In particular, Pursley and Frank investigated the use of the Reed-Solomon code and the Reed-Solomon/convolutional concatenated code (RS-CC) in [6]. For the Reed-Solomon code with no inner code, they used a $(32, 12)$ errors-only RS code with noncoherent reception and 27 codewords per packet. Thus, the code had rate $3/8$ and the total number of information bits per packet was 1620. There were 135 binary symbols per dwell period. For the concatenated code, they used a $(32, 18)$ RS code for the outer code and a rate $2/3$, constraint length 6 convolutional code with erasure threshold $\gamma = 3$ for the inner code. In addition, the convolutional code used soft decisions. The dwell interval spanned 159 binary symbols and there were 20 codewords per packet. The overall rate of the code was 3/8 and the number of information bits per packet was 1800. Requiring a PER of 10^{-3}, these results are shown in Figure 5.

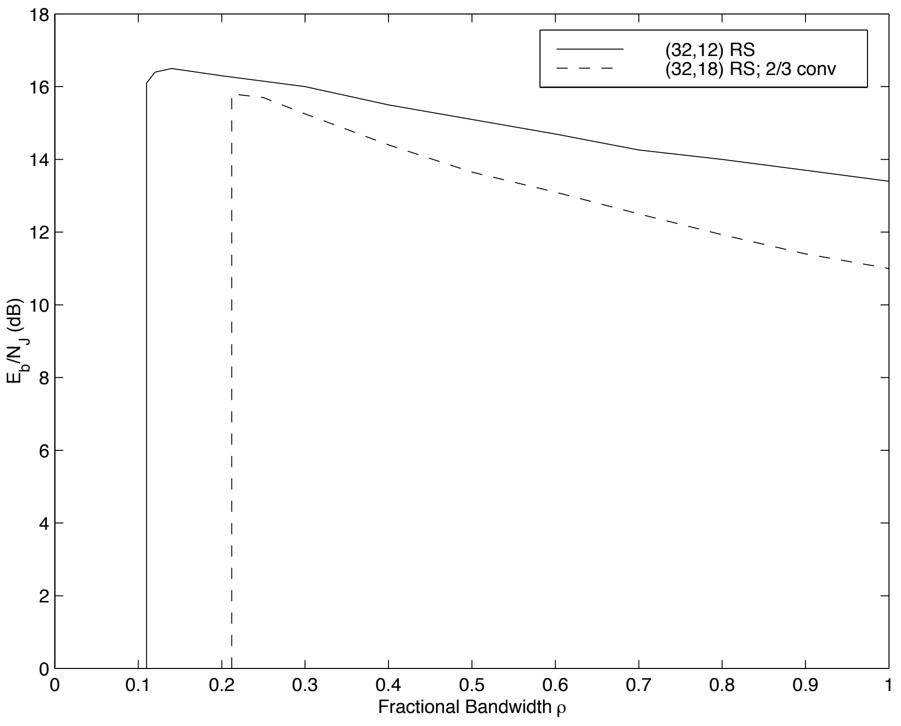

Figure 5: Other Codes with Noncoherent Reception

Because the turbo code system with 160 bits per hop matches the parameters shown in Figure 5 quite closely, we will use this system for comparison. Notice the large performance difference between the turbo code and the coding schemes of Figure 5. Comparing the worst-case performance of the turbo code system without side information with the RS and RS-CC systems, the turbo code system shows a gain of 6.9 and 6.3 dB, respectively. Note that to achieve worst-case performance for turbo codes, the jammer needs to jam the entire band ($\rho = 1$), while for RS and RS-CC codes, the jammer needs to jam only a small fraction of the band (about $\rho = .12$ and $\rho = .22$, respectively). Another performance measure of FH-SS systems is ρ^*, the minimum fractional jamming bandwidth required to induce any decoding errors. In this case, turbo codes save about .1 and .2 respectively.

While turbo codes seem to significantly outperform the RS and RS-CC codes, it is unfair to directly compare these results. First, the coding rates of the coding systems are different: 1/3 for the turbo code and 3/8 for the RS and RS-CC codes. However, this minor difference in code rate is not sufficient to explain the contrast in performance. Another difference is in the treatment of the memory case without side information. For turbo codes, we exploit the memory of the channel by estimating channel states and using this information in our branch transition probabilitiy calculations. For the RS-CC code, Pursley and Frank also use the memory to predict which hops have been jammed, but they do so in a very different way. The goal is to declare an erasure if the jamming in a dwell interval is sufficiently severe that many of the RS symbols are likely to be in error. The metric they use to estimate the channel is the Hamming distance between the binary code sequence chosen by the Viterbi decoder and the binary sequence that results from making hard decisions on the output of the demodulator. The resulting distance represents their estimate of the number of errors produced by the demodulator, γ. The different methods at which the turbo code and RS-CC systems treat the uncertainty of the channel state makes it difficult to compare the performance of the systems. The final major difference between the coding systems is that the turbo decoder is much more computationally complex. Even with the recursions of the MAP decoders, these calculations are iterated many times. Thus, we arrive at the familiar tradeoff between computational complexity and performance.

6 Numerical Results: Bounds

Figure 6 shows the numerical results for noncoherent reception when side information is available to the receiver. Note that the Bhattacharrya parameter, D, was calculated assuming worst case jamming. As shown in Figure 6, the average upper bound calculated for noncoherent reception is tight.

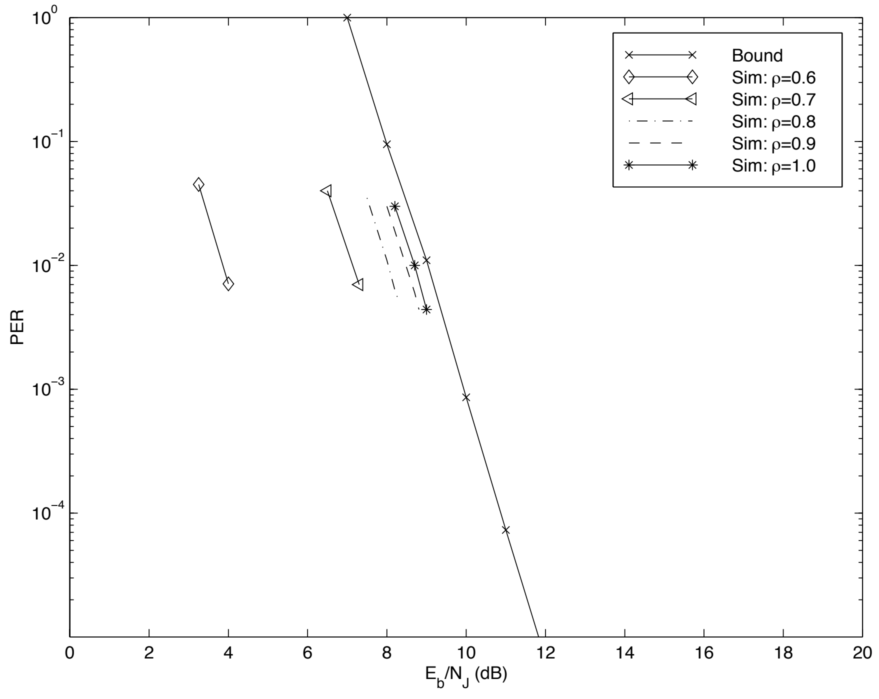

Figure 6: Bound: Noncoherent Reception, SI

7 Conclusion

We have shown that there exists great potential for noncoherently demodulated turbo codes in frequency-hop spread spectrum systems by analyzing cases with 1, 21, 81, and 159 bits per hop, and either with or without channel state side information. The case with 1 bit/hop and side information was uniformly superior relative to the other cases. We also witnessed performance degradation for the SI cases as the number of bits per hop increased due to the change in effective block length. However, as the length of the dwell period increased, performance differences between the SI and no SI cases diminished due to effective channel state estimation. Finally, we compared our simulation to other coding schemes and found the comparison to be favorable towards turbo codes.

While turbo codes are effective in reducing the E_b/N_J required to achieve a given packet error probability, it does so at great computational cost. Before turbo codes can be integrated into a packet radio network or any practical data communications system, its computational complexity must be reduced while minimizing performance losses. The work in this paper exemplifies the error correction power of turbo codes. Future research must investigate the application of lower complexity turbo decoders to FH-SS.

References

[1] J. Kang and W. Stark, "Turbo Codes for Coherent FH-SS With Partial Band Interference," *Proc. MILCOM '97*, November 1997.

[2] L. Bahl, J. Cocke, F. Jelinek, and J. Raviv, "Optimal decoding of linear codes for minimizing symbol error rate," *IEEE Trans. Information Theory*, vol. 20, March 1974.

[3] C. Berrou, A. Glavieux, and P. Thitimajshima, "Near Shannon limit error-correcting coding and decoding," *Proc. ICC '93*, May 1993.

[4] P. Robertson, "Illuminating the structure of code and decoder for parallel concatenated recursive systematic (turbo) codes," *Proc. GLOBECOM'94*, December 1994.

[5] W. Stark, "Coding for Frequency-Hopped Spread-Spectrum Communications with Partial-Band Interference - Part I: Coded Performance," *IEEE Trans. Communications*, vol. COM-33, October 1985.

[6] C. Frank and M. Pursley, "Concatenated coding for frequency-hop spread-spectrum with partial-band interference," *IEEE Trans. Communications*, vol. 44, no. 3, March 1996.

[7] M. Pursley and W. Stark, "Performance of Reed-Solomon coded frequency- hop spread-spectrum communications in partial-band interference," *IEEE Trans. Communications*, vol. COM-33, August 1985.

[8] S. Benedetto and G. Montorsi, "Design of Parallel Concatenated Convolutional Codes," *IEEE Trans. Communications*, vol. 44, May 1996.

[9] D. Divsalar, S. Dolinar, F. Pollara, and R. McEliece, "Transfer function bounds on the performance of turbo codes," *Telecom. and Data Aquisition Progress Report 42-122*, Jet Propulsion Laboratory, August 1995.

[10] C. Berrou, P. Adde, E. Angui, and S. Faudeil, "A low complexity soft-output Viterbi decoder architecture," *Proc. ICC '93*, May 1993.

[11] P. Robertson, E. Villebrun, and P. Hoeher, "A comparison of optimal and sub-optimal MAP decoding algorithms operating in the log domain," *Proc. ICC '95*, June 1995.

[12] S. Benedetto, D. Divsalar, G. Montorsi, and F. Pollara, "Algorithm for continuous decoding of turbo codes," *Electronics Letters*, vol. 32, no. 4, February 15, 1996.

[13] A. Barbulescu and S. Pietrobon, "Terminating the trellis of turbo codes in the same state," *Electronics Letters*, vol. 31, Nov. 1995.

[14] J. Mathis *et al*, "Final design plan: Sincgars packet switch overlay,", *SRI Tech. Rep.*, SRI Project no. 1244, July 1986.