Iterative Demodulation and Decoding for Protected Satellite Communications

Wayne G. Phoel

Military satellite communications (Milsatcom) systems use frequency hopping over a wide bandwidth to provide protection from hostile jamming, interception, and detection. These systems are being challenged to transport more information at higher rates, among smaller and more mobile terminals, with little or no increase in allocated bandwidth. Meeting these challenges requires advanced bandwidth-efficient and power-efficient signaling techniques. This article describes an approach to error-control coding and modulation that can help achieve the requirements of future Milsatcom systems. The receiver in this approach iterates between demodulation and decoding, which enables near-coherent performance with minimal reference symbol overhead. Also, the decoding process is augmented so that, in the presence of jamming, the receiver estimates the jammer state and combines information appropriately from different hops.

The need for protection from jamming, interception, and detection complicates the military satellite communications (Milsatcom) scenario significantly beyond the challenges encountered in commercial wireless communications systems. In spite of this fact, the defense community aims to duplicate in military environments the increased capabilities of commercial wireless devices available in the public domain. While there is some truth to the notion that protection comes at the cost of capacity, in general we can do much to provide bandwidth-efficient and power-efficient protected communications to the military end user. In particular, we can take advantage of recent advances in iterative decoding techniques to attain the desired level of protection with efficient use of bandwidth and power resources.

The scenario considered in this article concentrates on the jamming resistance of a frequency-hopping spread-spectrum system. Our approach uses frequency hopping combined with error-control coding to mitigate the effects of the jamming. In addition, by passing information on the reliability of symbols from the decoder back to the demodulator, and iterating several times, we can achieve near-coherent performance and also account for the effects of the jammer.

To resist hostile jamming, interception, and detection, protected Milsatcom systems communicate at high frequencies. The Milstar, Advanced EHF, and next-generation Transformational Satellite Communications (TSAT) systems use EHF and SHF frequencies at 44 GHz and 20 GHz, respectively. These high frequencies enable the use of small, highly directional antennas to minimize emissions away from the desired signal direction. These frequencies also come with relatively large bandwidths, which enable spread-spectrum techniques to hide the communications signal further as well as protect it from hostile attempts at disruption.

In frequency-hopping spread-spectrum systems, the transmitter hops from center frequency to center frequency according to a pseudo-random sequence, dwell-

FREQUENCY HOPPING FOR PROTECTION FROM JAMMING

FREQUENCY HOPPING allows the system to spread the communications signal over a wide bandwidth while keeping the instantaneous signal bandwidth processed by the transmitter and receiver relatively small. Both the transmitter and receiver compute the same pseudo-random sequence to determine on which frequency to communicate at a specific time. Typically, the transmitter remains at the center frequency for a fixed-length dwell interval and then hops to the next frequency. As a result, the term *hop* is often used to refer to the period the signal stays at one frequency.

The objective of the jammer is to disrupt the communications signal. The jammer, however, does not have unlimited transmit power. Therefore, its goal is to transmit a signal that will cause the most degradation of the desired signal, subject to its power constraint. The most common jammer strategy is a partial-band white-noise jammer. In this strategy, the jammer continuously transmits a Gaussian-distributed pseudo-random noise sequence with bandwidth a fraction ρ of the total bandwidth. Due to the power constraint, a narrow bandwidth (small ρ) corresponds to a large amount of energy concentrated in the communications signal instantaneous bandwidth. But a small ρ also decreases the likelihood that a frequency used to communicate gets jammed. Similarly, spreading the jamming signal over a wide bandwidth dilutes the jammer power and decreases the energy in a given signal band in order to increase the chances of degrading the communications. Therefore, from the perspective of a protected system, we want to force a jammer to operate over a large portion of the bandwidth, limiting the effectiveness of each hit by the jammer.

ing at each frequency for a fixed interval. The period of time during which the transmitted signal remains at that frequency is referred to as the *dwell interval*. The larger the total bandwidth used for hopping, the more difficult it is for the jammer to disrupt the communications. The sidebar entitled "Frequency Hopping for Protection from Jamming" provides more explanation of the concept of frequency hopping in a spread-spectrum system.

Iterative decoding, often referred to as turbo decoding, is becoming increasingly popular in communication system designs. Although iterative techniques had been used in systems before the advent of turbo codes, none of those techniques provided the dramatic gains that result from coupling the turbo-code design with the *a posteriori* probability (APP) soft-output algorithm used in the decoding process [1]. Since the introduction of turbo codes, iterative receiver designs have been applied to topics such as equalization, multiple user detection, and space-time coding [2]. The majority of research on such iterative receivers, however, assumes perfect knowledge of the side information (i.e., information that is not the ultimate goal of the receiver, but is used to decode the received signal, including parameters such as carrier phase, and signal and interference levels) necessary for the turbo decoder to achieve the channel capacity-approaching performance now expected of error-control codes.

In this article, we apply iterative demodulation and decoding to the slow frequency-hopping scenario in order to provide protection at low signal-to-interference ratios (the term *slow frequency hopping* means that more than one channel symbol is transmitted in each dwell interval). At the same time, we investigate techniques to estimate the relevant side information, such as the random carrier phase shift seen from hop to hop and the received signal reliability, including detection of jamming in dwell intervals.

Figure A illustrates the concept of frequency hopping. The hops are denoted by the shaded blocks. For this example we show only twelve hopping bands in the total frequency band. Ideally, the total bandwidth would consist of many hundreds of sub-bands to make the occurrence of jamming in dwell intervals appear independent from hop to hop.

In general, we model the jammer as a probability ρ that the center frequency of a given hop is jammed. If a hop is jammed, the power of the jamming signal in that hop is modeled as $N_J/2\rho$, where N_J is the average power spectral density of the jammer. If a small percentage of hops are jammed, we can correct the jammed symbols with error-control coding. But we must also ask how we quantify 'small.'

In the scenario considered in this article, the error-control coding takes the form of a block code (i.e., distinct blocks of information bits are encoded as separate blocks of code bits). In order for the code to perform well in a jamming environment, we want the incidence of jamming among code bits to appear independent. Not only do we want to spread the communications signal over a wide bandwidth, we want each code block to be spread over as many hops as possible. The number of hops over which the code block is spread is referred to as the *hop diversity*. In Figure A the hop diversity is 8. Therefore, for each hop that is jammed, 1/8 of the bits in a code block are affected (3/8 of the code block in Figure A is jammed).

Intuitively, if ρ is small, then there is a good probability that few or no hops of a given code block are jammed. If the hop diversity is large, then the error-control coding compensates for the jamming and the jammer is not effective. As ρ increases, however, the expected number of jammed hops per code block also increases. Increasing ρ also decreases the jammer power in a given sub-band. As a result, there is an optimum ρ at which the jammer causes the most degradation to the desired signal; for lower fractional bandwidths, too few hops are jammed to overwhelm the error-control code to the same degree, and for larger ρ, the jammer power becomes so diluted as to reduce its efficacy.

Performance of the desired signal generally improves by increasing the hop diversity, forcing the jammer to dilute its power over most, if not all, of the available bandwidth. However, with a fixed code block length (and with a requirement that all symbols in each hop come from the same code block) the only way to increase diversity is to shorten the number of symbols in a hop.

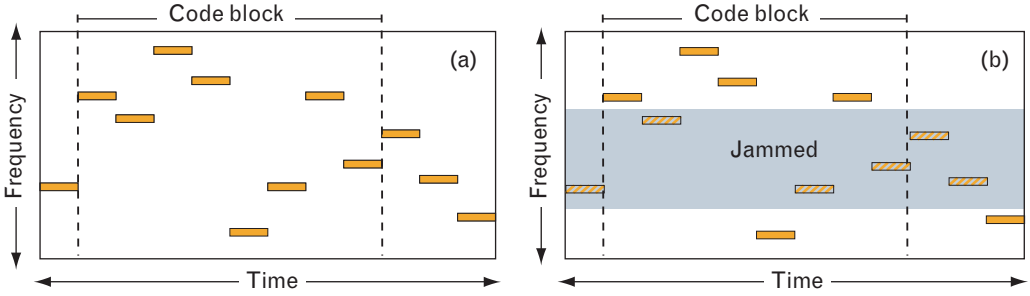

FIGURE A. Illustration of (a) unjammed frequency hopping in time and frequency space, and (b) frequency hopping in a region affected by a partial-band jammer. A frequency-hopping communications transmitter spreads the instantaneous narrowband signal over a wide bandwidth by hopping to a pseudo-random sequence of center frequencies that are known by the receiver. The partial-band jammer spreads its power over a fraction of the total hopping bandwidth in an attempt to disrupt communications. In this example, the jammer hits a frequency range roughly ten times the bandwidth of the instantaneous communications signal, thus jamming some of the dwell intervals in each transmitted code block. The receiver uses the more reliable information in the unjammed hops to improve the estimation of information in the jammed hops.

System Goals and Approach

In frequency-hopping systems, we cannot ensure carrier phase continuity at the boundaries of dwell intervals, and consequently we must reacquire the phase for each hop. For the purposes of this work, we assume the phase offset is constant for the duration of a dwell interval but it changes randomly from hop to hop. We might demodulate coherently by estimating the phase offset from known reference symbols transmitted in each dwell. This technique can be inefficient for short dwell intervals because the number of reference symbols required for an explicit estimate of the phase may result in excessive overhead, thus affecting both the power- and bandwidth-efficiencies. More typically, noncoherent schemes such as frequency-shift keying (FSK) or differential phase-shift keying (DPSK) are used to communicate with the unknown phase shift in each dwell. FSK, however, is much less spectrally efficient than we would like, and DPSK conventionally suffers by about 2 to 3 dB in required signal-to-noise ratio, compared to a coherent receiver.

In the frequency-hopping system considered here, we concentrate on DPSK with iterative demodulation and decoding. The number of reference symbols in this scenario is limited to one per dwell. As mentioned above, conventional demodulation of DPSK incurs a loss of approximately 2 dB. However, recent success in iteratively decoding interleaved concatenated codes shows excellent performance with a simple combination of a convolutional code, a pseudo-random permutation, and differential modulation [3]. This configuration can be considered a serially concatenated convolutional code (SCCC) in which the differential modulation acts as the inner code of the serial concatenation. Even when we are faced with an unknown phase offset, nearly coherent performance is often attainable [4, 5].

Of the iterative receiver techniques discussed in the literature, we focus on that described by M. Peleg, S. Shamai, and S. Galán, in which the continuously distributed random phase offset is approximated by a quantized phase [4]. An expanded trellis is then used to search over all possible sequences, assuming the now-limited number of possible phase offsets. The result is a joint estimation of the data and the carrier phase. Other techniques for dealing with unknown phase offset in an iterative fashion include averaging over the uniform random phase (resulting in the introduction of a Bessel function) in computing branch metrics in a trellis-based demodulator [5, 6], and iterative explicit phase estimation [7–9]. We note that averaging over the phase distribution results in degraded performance, compared to the receiver considered here, which provides nearly coherent performance in a straightforward manner.

For the scenario considered in this article, we limit the code block length to 2048 code bits (i.e., 1024 information bits assuming a rate-1/2 code) and require all symbols in a particular dwell to come from the same code block. Such a requirement limits the memory and processing needed at both transmitter and receiver. (Note: Internet protocol [IP] packets have an overhead of at least 480 bits, so that although a 1024-information-bit block may seem large, it is reasonable in a packet-based system.) The error-control code inserts redundancy into the transmitted sequence to enable error correction and/or detection. A rate-1/2 code doubles the number of bits used to represent the information sequence. For hops of short duration (e.g., one reference symbol and sixteen data symbols) we find that, with the technique considered here, there is significant loss with respect to coherent performance. However, having more symbols per hop translates to fewer hops over which a code block is spread, which compromises protection. Consequently, when jamming is introduced to the system model, there is a trade-off between protection from jamming and coherent demodulation. In that case, we find that having shorter dwells (and therefore more hops per block) results in significantly better performance in the worst-case partial-band jamming than having longer dwells.

The performance of turbo codes, or parallel concatenated convolutional codes (PCCC), in jamming is discussed by J.H. Kang and W.E. Stark [10, 11], wherein the trade-off between hop diversity and the duration of the hop was studied for coherent binary PSK (BPSK) [10] and M-ary FSK [11], with and without initial knowledge of whether a dwell interval is jammed. Those results assume that the receiver knows the jammer and noise power levels in order to scale the inputs to the APP decoders correctly. The decoding algorithm is sensitive to the scaling of the channel values; incorrect scaling may result in the failure of the iterative al-

gorithm to converge, especially in the presence of jamming. Yet accurately estimating the channel reliability can be difficult.

Results for turbo coding and jamming are also presented by M.A. Jordan for BPSK with perfect channel state information and ideal interleaving (i.e., the jammer state is independent from symbol to symbol) [12]. The effect of partial-band interference on the performance of turbo product codes is considered by M.B. Pursley and J.S. Skinner, where side information is used to erase symbols in dwells suffering from excessive interference [13]. With the exception of Reference 13, all the above references assume some knowledge of the background noise and jammer power spectra.

The results presented in this article require no initial knowledge of the signal-to-noise ratio for each dwell interval. Two aspects of side information are considered in addition to the phase offset. We first introduce a variation of the ratio-threshold test for FSK [14] to detect the presence of jamming for frequency-hopped PSK; then we apply that information to scale the inputs to the APP decoders appropriately [15]. The ratio-threshold test uses values naturally computed in the soft-output demodulator to revise the estimates of the jammer states at each iteration for each dwell interval.

Two similar approaches have been reported in the literature. Each explicitly estimates the noise variance in a dwell to scale channel values. In one approach, a clustering algorithm is used to distinguish jammed symbols from non-jammed symbols, and the noise variance is estimated separately for each group [16]. Iterating with the decoding is not performed. The other approach describes iterative noise variance estimation in combination with a PCCC and with a convolutional code [17]. There the variance estimates are improved throughout the iterations, based on the *a priori* information generated by the decoder. The performance presented here is comparable to that for the turbo code [17], but requires a less complex receiver.

Signal Model

Figure 1 shows a block diagram of the frequency-hopping system under consideration. A block of information bits is first appended with zeros to terminate the code in the zero state and is then encoded with a binary convolutional code. The resulting sequence of code bits is permuted and Gray-mapped onto an M-ary phase-shift key (MPSK) constellation

$$\left\{ e^{j2m\pi/M} \,\middle|\, 0 \le m < M \right\}.$$

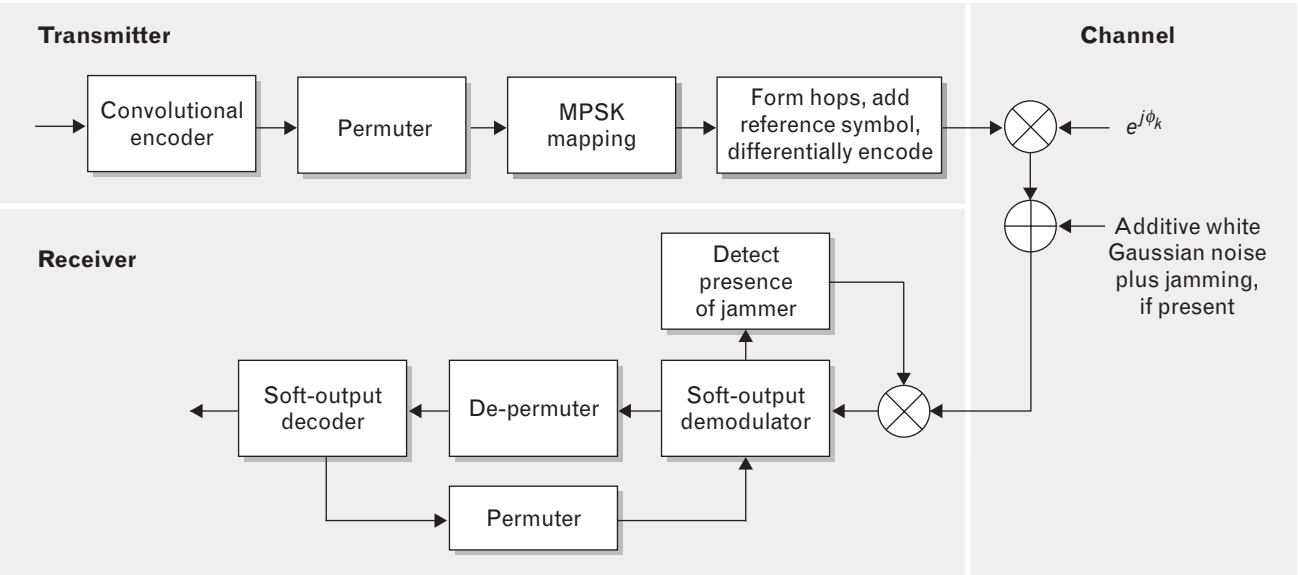

FIGURE 1. Block diagram of the frequency-hopping system. The transmitter encodes a block of data with a simple convolutional code. The order of the code bits is then permuted and differentially modulated by using M-ary phase-shift keying (MPSK). The channel introduces a random phase shift on each hop, adds noise, and adds a jamming signal, if present. The receiver iteratively demodulates and decodes the received signal, jointly estimating the data and the channel state.

The permutation is fixed but computed according to a pseudo-random process. The randomness of the permutation effectively distributes the information contained in each bit throughout the code word, improving the error-correcting capability of the code. This improvement is dramatic when the decoder iterates between demodulation and decoding. The permutation also decorrelates the presence of the jammer in consecutive bit estimates as seen by the outer convolutional code. Gray mapping assigns a sequence of $\log_2 M$ bits to one of M constellation points such that only one bit differs in sequences mapped to adjacent symbols. For example, a valid Gray mapping of all length-two binary sequences to a quaternary constellation maps (0, 0) to 1, (0, 1) to j, (1, 0) to $-j$, and (1, 1) to –1.

The single complex value used to represent the M bits for transmission over the channel is called a *channel symbol*. The sequence of MPSK channel symbols is parsed into equal-length subsequences, which form the signals to be transmitted in the dwell intervals. Each interval begins with an arbitrarily chosen reference symbol known at both the transmitter and receiver. The data-bearing channel symbols form the remainder of the transmission for the dwell. Let L be the number of channel symbols transmitted in each dwell interval. If we use a discrete-time baseband system model, the lth channel symbol transmitted in the kth dwell is given by

$$x_{k,l} = x_{k,l-1} u_{k,l} ,$$

where $u_{k,l}$ is the lth Gray-coded M-ary symbol prior to differential encoding, and

$$x_{k,0} \in \left\{ \sqrt{2} e^{j(2m+1)\pi/M} \right\}$$

is deterministic.

This combination of convolutional coding with pseudo-random permutation and differential modulation is essentially an SCCC in the context of a slow frequency-hopping system. The SCCC consists of two constituent codes: the outer binary convolutional code and the inner differential modulation (the differential MPSK can be represented as an M-ary convolutional code). For the results presented here, we consider only $M = 4$, but it is straightforward to extend the technique to other values of M.

Each dwell interval is assumed to be transmitted at a frequency chosen according to a pseudo-random sequence known at both the transmitter and the receiver. In our baseband model, the only evidence of the frequency hopping is an independent phase offset for each hop and the independence of the jamming among different hops. The phase offset is assumed to be constant for the duration of a dwell and varies randomly from interval to interval according to a random variable distributed uniformly between 0 and 2π radians. The received signal is also corrupted by thermal additive white Gaussian noise (AWGN) and, in the presence of partial-band jamming, an additional AWGN term representing the jamming signal. The corresponding received signal can be written as

$$y_{k,l} = A_k x_{k,l} e^{j\phi_k} + z_{k,l} + \nu_k w_{k,l} ,$$

where A_k is the gain of the data in the received signal in dwell interval k, ϕ_k is the phase offset for dwell k, $z_{k,l}$ represents the complex-valued background noise with two-sided power spectral density $N_0/2$, ν_k is a Bernoulli random variable taking the value 1 if partial-band jamming is corrupting the hop and 0 if not, and $w_{k,l}$ is the jamming signal with average two-sided power spectral density $N_J/2$.

The jammer is characterized by the fraction ρ of the band that is jammed. We model this as the probability ρ that a particular hop is jammed (i.e., $\nu_k = 1$ with probability ρ), where we assume that if a hop is jammed, the full duration and bandwidth of that hop is jammed. Therefore, the power spectral density of the jammer, when present, is $N_J/2\rho$.

Iterative Receiver

The demodulator uses a soft-output trellis-based algorithm based on APP decoding, and operating in the log domain. The appendix entitled "Turbo Decoding and the APP Algorithm" provides a more detailed description of APP decoding. Since the carrier phase is unknown, we approximate the continuously distributed random phase as a random variable with a discrete distribution [4, 18]. The number of phases in the discrete distribution is set to $8M$, thus requiring a trellis with $8M$ states (rather than only M states in the case of known phase) for demodulation. The state of the signal at a specific time is given by the current phase of

the phase-shifted channel symbol (i.e., $x_{k,l}e^{j\phi_k}$). Therefore, for differentially encoded quaternary PSK (QPSK) the quantized phase states comprise the values $e^{js\pi/16}$, where $s = 8m + n$, in which m represents the quaternary symbol ($0 \le m \le 3$) and n corresponds to the quantized phase offset modulo $\pi/2$ ($0 \le n \le 7$). The eight sets of rotated QPSK constellations therefore form cosets. Figure 2 uses eight different symbols to depict these cosets. The demodulator produces soft estimates of the code bits one dwell interval at a time. Since the phase is assumed to be constant over the duration of the dwell, there can be transitions only between states that are in the same coset.

After the symbols are demodulated in all dwells and the reference symbols are stripped, the code bit estimates are de-permuted and sent to the soft-output decoder. This outer decoder also uses a log-domain APP algorithm and generates extrinsic information (that is, information about a bit's value derived from adjacent bits in the sequence) on the code bits, which is passed back to the demodulator. The demodulation and decoding continue in an iterative fashion until the stopping criterion is met (the criterion used here is a fixed number of iterations). The basic APP algorithms used for demodulation and decoding are the same as those commonly used in the literature [19, 3, 4]. The two

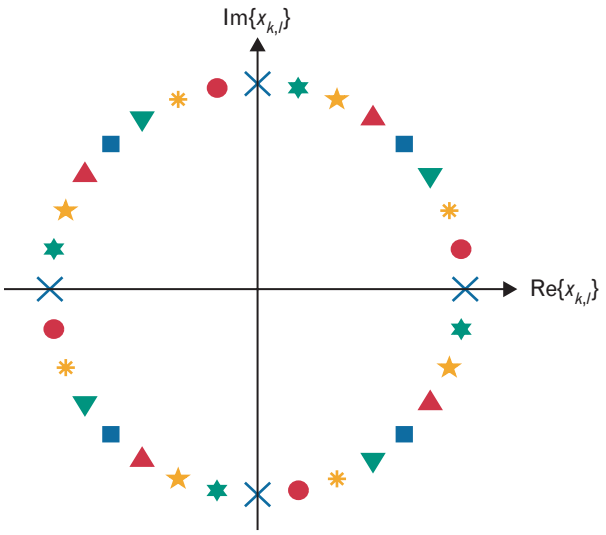

FIGURE 2. Eight cosets for demodulating differentially encoded quaternary phase-shift key (QPSK) constellations. The four points shown with a particular symbol indicate the rotated QPSK constellation corresponding to a specific hypothesized phase offset.

main differences are that the demodulation is done one dwell interval at a time and the results of the forward recursion in the demodulator are used for detection of jamming. For that reason, and to introduce notation, the forward recursion is described here in more detail.

During the forward recursion of APP demodulation, with each input symbol the demodulator computes a statistic related to the joint probability for each state (see, for example, [19]),

$$q_{k,l}(s) = \max_{s'}{}^{*}[q_{k,l-1}(s') + c_{k,l}(s', s)],$$

where $q_{k,l}(s)$ corresponds to state s for the lth channel symbol in dwell interval k, $\max^*(a, b) = \ln(e^a + e^b)$ is the Jacobian logarithm [20], and $c_{k,l}(s', s)$ is the log-domain probability statistic corresponding to a transition from state s' to state s and includes the log-likelihood ratio of $y_{k,l}$ as well as *a priori* information from the outer decoder [in the terms used in the appendix for describing APP decoding, $c_{k,l}(s', s)$ is related to the logarithm of $\gamma_k(s', s)$]. A similar process occurs for the backward recursion. For the max-APP decoder, the Jacobian logarithm is replaced by the $\max(\cdot,\cdot)$ operation. The relationship between $q_{k,l}(s)$ and the joint probability of the state s and all previous channel values

$$\alpha_{k,l}(s) = \Pr(s, \mathbf{y}_0, \mathbf{y}_1, \ldots, \mathbf{y}_{k-1}, y_{k,0}, y_{k,1}, \ldots, y_{k,l}),$$

is given by

$$\alpha_{k,l}(s) = \frac{\exp[q_{k,l}(s)]}{\sum_{s} \exp[q_{k,l}(s)]},$$

where $\mathbf{y}_k$ denotes the sequence

$$\{y_{k,0}, y_{k,1}, \ldots, y_{k,L-1}\}.$$

The typical log-APP decoder assumes the inputs are log-likelihood ratios and therefore requires knowledge of the received signal magnitude and noise power (and jammer power, if present). In particular, the log-likelihood ratio input to the demodulator is $L_k y_{k,l}$, where L_k, known as the *channel reliability factor*, is a function of the signal amplitude and the total noise power in interval k. For optimal performance,

$$L_k = \frac{4A_k}{N_T},$$

where

$$N_T = N_0 + \nu_k N_J/\rho \ .$$

Consequently, the magnitude of the desired component of the log-likelihood ratio is proportional to the signal-to-noise ratio in the dwell. When a jamming signal is present in a dwell interval, the optimal input to the demodulator is $L_J y_{k,l}$, where $L_J = 4A/(N_0 + N_J/\rho)$, and we assume $A_k = A$ for all hops; in the absence of jamming, the channel value is scaled by $L_N = 4A/N_0$. If the input is a log-likelihood ratio, the output will also be a log-likelihood ratio and can be passed to the outer decoder, but with the *a priori* information removed.

Performance in Additive White Gaussian Noise with No Jamming

This first set of simulation results illustrates how the iterative receiver provides significant gains as the number of iterations increases. For all results presented here, the block size is limited to 1022 information bits (plus two terminating bits). The convolutional encoder is non-recursive, has memory 2 and rate 1/2, and uses the generating polynomial specified in octal form as (7, 5). Figure 3 shows a diagram of the convolutional encoder. An s-random interleaver of length 2048 is used to permute the code bits before they are modulated with differentially encoded QPSK.

Figure 4 shows the bit error rates of the SCCC for 1, 2, 5, 10, and 20 iterations when the receiver has perfect knowledge of the carrier phase for all symbols. In fact, in this scenario the signal is not hopped and there is no jamming. The energy per bit, E_b, is defined as the average received energy per transmitted information bit. Therefore, it includes the additional energy due to coding overhead as well as that from reference symbols. The

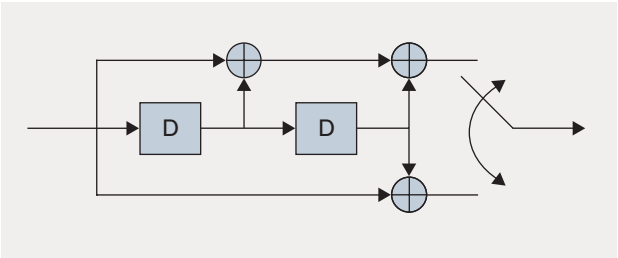

FIGURE 3. Linear shift register implementation of the convolutional encoder. All addition is modulo 2.

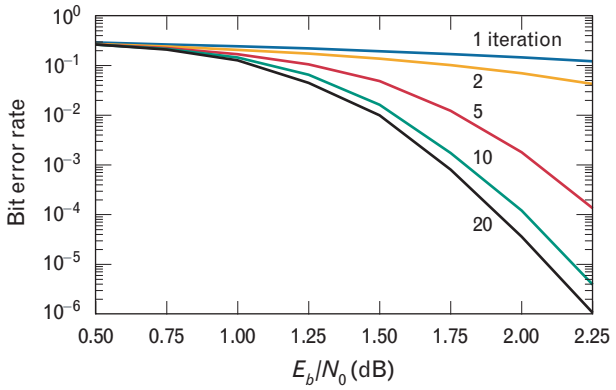

FIGURE 4. Performance of iterative demodulation and decoding of QPSK, with no frequency hopping. Perfect side information (phase offset, amplitude, and noise power) is assumed. The curves illustrate the turbo-like code performance achieved throughout the iterations.

scaling factor applied to the received symbols to produce log-likelihood values is given by $L_N = 4A/N_0$, and we assume that the signal amplitude and noise power are known perfectly. For ten or more iterations the bit error rate decreases by more than an order of magnitude for each 0.25 dB increase in E_b/N_0. It has been shown that this behavior can be attributed to the combination of the long pseudo-random permutation and the recursive inner code [21]. Note that the theoretical channel capacity for a code block length of 2048 and a rate-1/2 code requires approximately 1 dB E_b/N_0, so this simple code performs approximately 1 dB from the capacity of the AWGN channel. The gain from iterating the receiver begins to decrease around ten iterations. For all subsequent scenarios, ten iterations of the receiver are performed.

Figure 5 shows how the performance of iterative demodulation and decoding with unknown phase offset approaches that of a coherent system as the dwell interval increases. Results are shown for dwell intervals of L = 1025, 65, and 17 symbols per hop. The reference symbols increase E_b by 0.07 dB for L = 65 and by over 0.26 dB for L = 17. The coherent performance shown here corresponds to the ten-iteration curve in Figure 4. For L = 1025 the curve differs from the coherent case by less than 0.1 dB. The scenario with L = 65 performs within 0.25 dB of the coherent system. However, the loss becomes increasingly significant as the hop size is decreased to only 17 symbols per hop. In that scenario the required E_b/N_0 increases by approximately 0.7 dB.

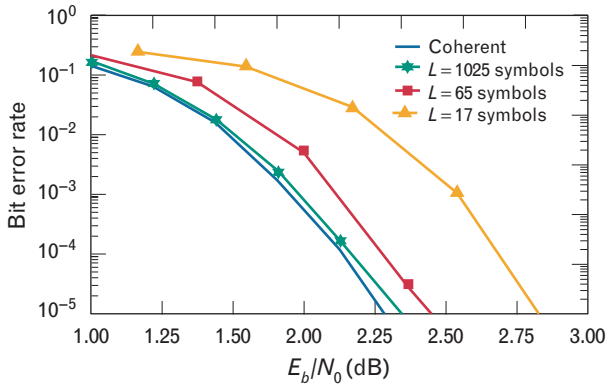

FIGURE 5. Performance of iterative decoding in full-band additive white Gaussian noise with unknown phase offsets and no jammer; L = 1025, 65, and 17 channel symbols per dwell. The performance loss is due to the additional energy per bit from the increased reference symbols per block and the imperfect carrier phase synchronization for hops with relatively few symbols.

(Note that this increase in required E_b/N_0 includes the energy in the additional reference symbols.)

Clearly, in the absence of partial-band jamming, long dwell intervals are preferable to shorter dwells for the coding and modulation considered here. In addition to increasing the energy in reference symbols, the shorter dwells provide less information about the phase offset and therefore do not approach coherent performance as closely as do the longer dwells.

Performance in Jamming with Knowledge of the Jammer State

In this section we present simulation results that illustrate the effect of partial-band white-noise jamming on the performance of the system. As described above, the partial-band jammer is modeled as a probability ρ that a dwell interval is jammed, and the jammer power is N_J/ρ in such an interval. For the simulation of jamming, $E_b/2N_J$ is varied with E_b/N_0 fixed at 10 dB. Also, the energy in the reference symbols is included when we compute E_b for the figures.

We assume that the receiver knows which hops are jammed and adjusts the log-likelihood ratios accordingly. Figure 6 compares the performance as a function of the fractional bandwidth jammed, ρ, for a target bit error rate of 10^{-4}. That is, for a particular ρ and for E_b/N_0 = 10 dB, the figure shows the required E_b/N_J to achieve a bit error rate of 10^{-4}. The points at ρ = 1 correspond to the required E_b/N_0 for the desired bit error rate in Figure 5, shifted to account for separate background noise and jamming signal powers. The degradation of the L = 65 symbol dwell interval is pronounced, requiring an additional 1.5 dB E_b/N_J at the worst-case fractional bandwidth than for a full-band jammer. Having only sixteen hops in a code block does not provide sufficient hop diversity for protection from jamming. Of the two dwell intervals compared here, L = 65 performs the best in Figure 5 (by over 0.5 dB) and for $\rho > 0.82$ in Figure 6. However, Figure 6 shows that L = 17 performs more consistently over the range of ρ and it has the best performance for the worst-case partial-band jammer. At an error rate of 10^{-5}, simulation results show that the loss for L = 65 with respect to L = 17 exceeds 2 dB (this result is not shown here). A similar trade-off was reported elsewhere for a fixed block size, wherein the longer hop length was used to provide better side information indicating whether the hop was jammed or not [11].

The performance of iterative decoding can be sensitive to errors in the channel reliability factors used. However, in actuality, the receiver does not know initially what the noise or jammer power spectral densities are, nor does it know which scaling factor to use in a particular dwell interval. For similar decoding schemes without partial-band jamming, a major concern is that

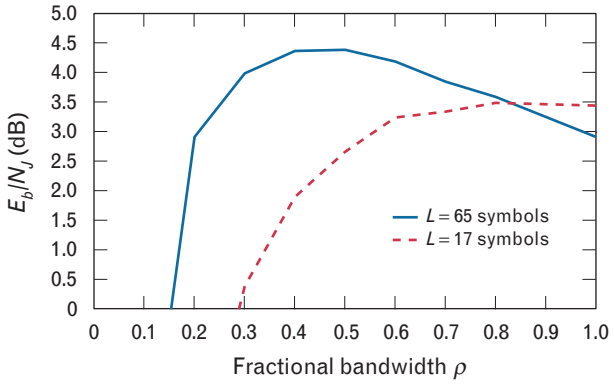

FIGURE 6. Required E_b/N_J as function of the fractional bandwidth ρ for 10^{-4} bit error rate; 10 iterations; L = 65, 17. The worst-case partial-band jammer fills about half of the available bandwidth for the 65-symbol-hop configuration, and drives up the required desired signal power by 1 dB with respect to the worst case with 17 symbols per hop. The reason for this difference in performance is the increased hop diversity of L = 17 symbol hops (diversity of 64 hops per code block) compared to L = 65 (only 16 hops per code block).

the channel symbols and extrinsic information are scaled appropriately between decoders. The desired performance can be obtained by setting L_N so that it is optimal at the target error rate. Yet with the presence of partial-band jamming we not only need to ensure that the *a priori* information and the channel symbols are weighted correctly, but we must also weight bits from dwells containing jamming in the appropriate proportion to those without jamming. Of course, first we need to identify those symbols which are jammed and those which are not.

Ratio-Threshold Test for Phase-Shift Keying

The technique considered here is based on the ratio-threshold test for FSK introduced by A.J. Viterbi [14]. For FSK, this test compares the two largest outputs of the envelope detectors to determine whether a received symbol contains excessive interference. The largest detector output is assumed to contain the desired signal plus noise and jamming, if present. The second largest output is intended to represent the interference level. If the ratio of the largest value to the second largest value is greater than some predetermined threshold, then we conclude that no significant interference is present. If the ratio is less than the threshold, then we decide that the signal is dominated by interference and we declare that jamming is present.

One of the keys to Viterbi's test is that the statistics being compared are from orthogonal filters. Unfortunately, because of the spectral efficiency of our system, we do not have orthogonal filters to generate independent statistics. To apply a ratio-threshold test for PSK, we make use of the expanded trellis used to demodulate the differentially encoded signal. At the end of the forward recursion for dwell interval k, the log-domain routine outputs the joint statistic for each state $q_{k,L-1}(s)$. In general, the trellis states corresponding to the quantized phase that is closest to the true phase offset will produce the greatest joint probabilities and therefore the largest values for $q_{k,L-1}(s)$. In addition, the quantized phase that is halfway between adjacent phases from the maximizing coset will contain a greater proportion of the jamming signal. For QPSK, this is a shift of $\pi/4$ radians. No output of the demodulator can be assumed to contain only noise.

Whereas the ratio-threshold test for FSK is performed on individual channel symbols, this version for PSK detects jamming over a whole hop. We define

$$s_k^* = 8m^* + n^*$$

as the state that maximizes $q_{k,L-1}(s)$. The important parameter here is n^*, which specifies the coset with the maximum joint probability. Therefore, the coset having the greatest proportion of jamming is specified by

$$\tilde{n} = (n^* + 4)_{\mathrm{mod}\,8}$$

and the state used for comparison with s_k^* is given by

$$\tilde{s}_k = \arg\max_{s:n=\tilde{n}} q_{k,L-1}(s).$$

Then the statistics we wish to compare to detect jamming in dwell k are $q_{k,L-1}(s_k^*)$ and $q_{k,L-1}(\tilde{s}_k)$. Left unchanged, $q_{k,L-1}(s_k^*)$ and $q_{k,L-1}(\tilde{s}_k)$ are correlated because of two components: the presence of the signal component and the *a priori* information from the previous iteration.

Decision Statistic

To start, we consider the first iteration of the decoder. Since the receiver has no initial knowledge of the state of the jamming in any of the dwells, L_k is initialized to be L_N for all k. (Forming a ratio of the two statistics helps alleviate the effects of incorrectly scaling the symbols containing jamming.) Let $q_k^*(i)$ and $\tilde{q}_k(i)$ denote the signal and jamming statistics for the ith iteration, respectively. If the path through the trellis for $\tilde{q}_k(0)$ corresponds to the correct code sequence, then

$$\tilde{q}_k(0) \geq \sum_{l=0}^{L} \mathrm{Re}\left\{e^{-j\pi/4} x_{k,l}^* y_{k,l}\right\},$$

where here the superscript * denotes the complex conjugate and the inequality is due to the max* operation. (Consequently, with max-APP decoding, if the path is correct, this expression becomes an equality.) If the corresponding path through the trellis is not the correct code word then this is a looser bound and the inequality still holds. The term in the summation in the above equation can be rewritten as

$$\frac{\sqrt{2}}{2}\left[\mathrm{Re}\left\{x_{k,l}^* y_{k,l}\right\} + \mathrm{Im}\left\{x_{k,l}^*(z_{k,l} + v_k w_{k,l}\right\}\right].$$

Similarly, we determine the lower bound of $q_k^*(0)$ as

$$q_k^*(0) \geq \sum_{l=0}^{L} \mathrm{Re}\left\{x_{k,l}^* \, y_{k,l}\right\},$$

if that path corresponds to the correct code word. As the iterations progress, neglecting the *a priori* information, the bound becomes tighter. Therefore, we can reasonably make the following approximation:

$$\sqrt{2}\tilde{q}_k(0) - q_k^*(0) \approx \sum_{l=0}^{L} \mathrm{Im}\left\{x_{k,l}^*(z_{k,l} + v_k w_{k,l})\right\},$$

which no longer contains a component due to the desired signal. Note that for larger values of M, this approach—while still valid—is more complicated and suffers from degraded performance.

In later iterations we need to remove the *a priori* information. This extrinsic information helps choose the right sequence through the trellis for detecting the jammer, but this additional information should not be included in the statistic used in the ratio test (otherwise, as the extrinsic information grows it will dominate all other terms and the ratio will approach one, regardless of the jammer state). One of the goals of this work is to avoid unnecessarily increasing the complexity of the receiver. Since the total contribution of the *a priori* information is not a direct by-product of the APP algorithm used, we use an *ad hoc* technique to approximate the contribution of the extrinsic information by using parameters already computed in the demodulator.

In particular, we use the growth of $q_k^*(i)$ throughout the iterations as a measure of the growth of the *a priori* values, and we subtract that from both the signal and jamming statistics. Let

$$\Delta q_k(i) = q_k^*(i) - q_k^*(0).$$

Then the signal statistic at iteration i remains

$$q_k^*(0) = q_k^*(i) - \Delta q_k(i)$$

and the jamming statistic at iteration i becomes

$$\hat{q}_k(i) = \sqrt{2}\left[\tilde{q}_k(i) - \Delta q_k(i)\right] - q_k^*(0).$$

The ratio to detect jamming in hop k at iteration i is

$$\eta_k(i) = \frac{q_k^*(0)}{\hat{q}_k(i)}.$$

The threshold that experimentally performs the best is 4.0. Therefore, in the results presented here we decide hop k is jammed if $\eta_k(i) < 4.0$, and it is declared jammer-free if $\eta_k(i) \geq 4.0$.

Scaling of Channel Symbols

The next problem to consider is how to weight the channel values from different dwells when we have estimated the presence of jamming but we do not know the correct channel reliability factors. One solution is to estimate these values. However, limiting the number of reference symbols available has rendered conventional estimation techniques ineffective. Alternatively, we can erase jammed symbols. The problem with this approach, though, is that large values of ρ often cause us to erase too many symbols so that the decoder does not have enough information to estimate the transmitted code word.

To avoid erasing too many symbols, we propose erasing them only if they do not amount to a significant portion of the code word. We do want to have some effect on blocks that contain many jammed hops; therefore, rather than erasing jammed symbols, we scale back on the reliability factor. To do so, we estimate the fraction of the band that is jammed, or more accurately the fraction of dwells in the code word that are jammed, and use this estimate, $\hat{\rho}$, to scale the channel symbols. For the case in which no jamming is detected in the dwell interval, we use a scaling factor corresponding to $E_b/N_0 = 7$ dB. When jamming is detected we assume a scale factor corresponding to $E_b/N_T = 3$ dB and then decrease the reliability factor by using the estimate $\hat{\rho}$. We chose $E_b/N_T = 3$ dB because it is slightly greater than the signal-to-noise ratio at which the code achieves the desired bit error rate. If only one hop in a code block is detected with jamming, symbols from that block are erased. If more than one jammed hop is detected, the symbols are scaled by $\hat{\rho}^2$ (which gives improved performance at low values of ρ compared to scaling by $\hat{\rho}$).

Estimated Jammer State

The required E_b/N_J as a function of the jammer fractional bandwidth when the channel reliability factor is based on the estimate $\hat{\rho}$ is illustrated in Figure 7. There is a minimal degradation in performance for $L = 65$ due to detecting jamming and estimating L_J rather than us-

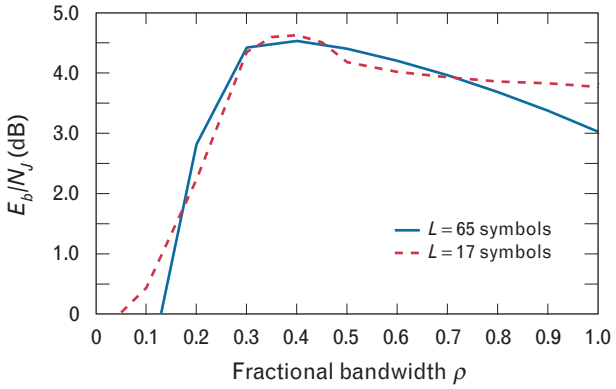

FIGURE 7. Required E_b/N_J as function of ρ for estimated jamming detection; with unknown N_T; weighted scaling of jammed inputs through $\hat{\rho}$; 10 iterations; $L = 65, 17$. The additional symbols in the 65 symbol hops enable accurate detection of and reaction to jamming. However, the relatively few symbols in the 17 symbol hops result in significant degradation with respect to performance when the jammer state is known perfectly.

ing perfect knowledge of the jammer state. The $L = 65$ curve in Figure 7 is about 0.1 dB worse than the corresponding curve in Figure 6. This scaling technique has quite a different effect on the performance when $L = 17$. The range of ρ between 0.2 and 0.45 exhibits a significant loss. A threshold lower than 4.0 works better in this range, but produces much worse results for lower values of ρ.

To evaluate the performance of this ratio-threshold test for detecting partial-band white-noise jamming of our signal design, we compare the probability of false alarm P_{FA} (i.e., the hop was not jammed, but the ratio did not exceed the threshold), and the probability of detection P_D for a fixed E_b/N_J. We consider two degrees of jammer state information: one in which we know the background noise and jammer level (referred to as "known N_T") and scale the channel values accordingly, and one in which we assume no knowledge of N_0 or N_T in scaling channel values.

Figure 8 shows the false-alarm probability, the probability of a missed detection $(1 - P_D)$, and the bit error rate for each iteration, with $L = 17$, $E_b/N_J = 4.0$ dB, and $\rho = 0.6$. For the curves corresponding to unknown N_T, the inputs to the decoder are scaled by $\hat{\rho}^2$ as described above. Results for $L = 65$ are not shown because there were almost no missed detections for any of the simulations and only a significant number of false alarms for the first few iterations.

Figure 8 shows that the probability of false alarm and the probability of a missed detection each improve as the bit error rate improves. The bit error rates for both receivers are nearly identical for the first few iterations, but the bit error rate of the receiver for unknown N_T pulls away from that for known N_T around the third or fourth iteration. Interestingly, for the first several iterations, the P_{FA} is less for known N_T than for unknown N_T. In fact, the P_{FA} for unknown N_T does not reach below that for known N_T until after the two bit error rates have diverged significantly. This observation suggests that the improvement in false-alarm probability throughout the iterations is an effect, rather than a cause, of the improved bit error rate. However, the probability of a missed detection is consistently better for the unknown N_T receiver. Consequently, for this comparison, the probability of detecting jamming is more important than P_{FA}. Weighting jammed signals as much as unjammed signals when the jammer is much stronger than the noise causes errors to propagate through the code block as the receiver performs its iterations. False alarms do not have such a catastrophic effect.

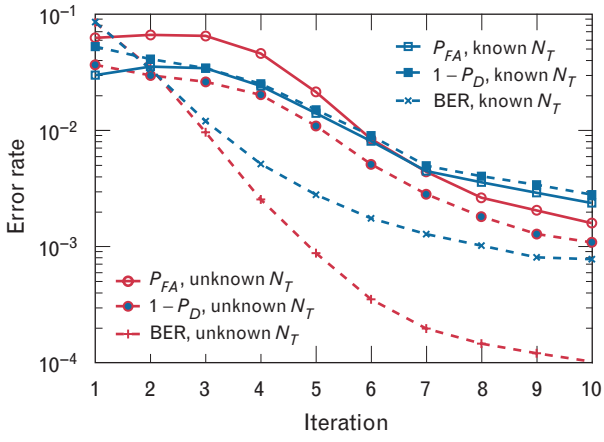

FIGURE 8. Probabilities of false alarm (P_{FA}), missed detection ($1 - P_D$), and bit error rate (BER) through ten iterations for known and unknown N_T, ratio-threshold test, $L = 17$, $E_b/N_J = 4.0$ dB, and $\rho = 0.6$. Although for earlier iterations, P_{FA} is better when N_T is known than when it is not known, BER starts to become better for the scenario of unknown N_T. On the other hand, $(1 - P_D)$ is better for unknown N_T at each iteration, implying detection probability plays a greater role in decoder performance than the false-alarm rate.

Conclusion

This article discusses a signal design for reliable protected communication in the presence of jamming. In addition, we introduce a ratio-threshold test for PSK for a frequency-hopped communications system suffering from partial-band white-noise jamming. With only one reference symbol per dwell, the receiver must use the data-bearing symbols in the hop to correct for the random phase experienced on each hop as well as to detect the presence of jamming. The ratio-threshold test estimates the presence of jamming in a hop. That information is then used to scale the channel values in the dwell interval appropriately for use in iterative APP demodulation and decoding.

The ratio-threshold test detects jamming well, with both the probability of false alarm and the probability of detection improving throughout the iterations. Of the two dwell interval lengths considered, that with more hops per code block performs the best when perfect knowledge of the jammer state is available. However, when the receiver must estimate this side information, both configurations perform similarly, with a modest gain from having more symbols in a hop. Clearly, more symbols per dwell are preferable from the perspective of estimating jamming and exploiting phase coherence.

REFERENCES

1. C. Berrou and A. Glavieux, "Near Optimum Error Correcting Coding and Decoding: Turbo-Codes," *IEEE Trans. Commun.* **44** (10), 1996, pp. 1261–1271.
2. K. Chugg, A. Anastasopoulos, and X. Chen, *Iterative Detection: Adaptivity, Complexity Reduction, and Applications* (Kluwer, Boston, 2001).
3. K.R. Narayanan and G.L. Stuber, "A Serial Concatenation Approach to Iterative Demodulation and Decoding," *IEEE Trans. Commun.* **47** (7), 1999, pp. 956–961.
4. M. Peleg, S. Shamai (Shitz), and S. Galan, "Iterative Decoding for Coded Noncoherent MPSK Communications over Phase-Noisy AWGN Channel," *IEE Proc. Commun.* **147** (2), 2000, pp. 87–95.
5. G. Colavolpe, G. Ferrari, and R. Raheli, "Noncoherent Iterative (Turbo) Decoding," *IEEE Trans. Commun.* **48** (9), 2000, pp. 1488–1498.
6. M. Peleg and S. Shamai (Shitz), "Iterative Decoding of Coded and Interleaved Noncoherent Multiple Symbol Detected DPSK," *Electron. Lett.* **33** (12), 1997, pp. 1018–1020.
7. P. Hoeher and J. Lodge, "'Turbo DPSK': Iterative Differential PSK Demodulation and Decoding," *IEEE Trans. Commun.* **47** (6), 1999, pp. 837–843.
8. L. Zhang and A. Burr, "Phase Estimation with the Aid of Soft Output from Turbo Decoding," *Proc. 2001 IEEE Vehicular Technology Conf.* **1,** *Atlantic City, N.J., 7–11 Oct. 2001,* pp. 154–158.
9. A. Anastasopoulos and K.M. Chugg, "Adaptive Iterative Detection for Phase Tracking in Turbo Coded Systems," *IEEE Trans. Commun.* **49** (12), pp. 2135–2144.
10. J.H. Kang and W.E. Stark, "Turbo Codes for Coherent FH-SS with Partial Band Interference," *Proc. IEEE 1997 Military Communications Conf.* **1,** *Monterey, Calif., 3–5 Nov. 1997,* pp. 5–9.
11. J.H. Kang and W.E. Stark, "Turbo Codes for Noncoherent FH-SS with Partial Band Interference," *IEEE Trans. Commun.* **46** (11), 1998, pp. 1451–1458.
12. M.A. Jordan, "Turbo Code Performance in Partial-Band Jamming," *Proc. 1998 IEEE Military Communications Conf.* **3,** *Bedford, Mass., 19–21 Oct. 1998,* pp. 982–986.
13. M.B. Pursley and J.S. Skinner, "Turbo Product Coding in Frequency-Hop Wireless Communications with Partial-Band Interference," *Proc. 2002 IEEE Military Communications Conf.* **2,** *Anaheim, Calif., 7–10 Oct. 2002,* pp. 774–779.
14. A.J. Viterbi, "A Robust Ratio-Threshold Technique to Mitigate Tone and Partial Band Jamming in Coded MFSK Systems," *Proc. 1982 IEEE Military Communications Conf.* **1,** pp. 22.4.1–22.4.5.
15. W.G. Phoel, "Side Information for Iterative Demodulation and Decoding of Frequency-Hopped PSK in Partial-Band Jamming," *Proc. 2003 IEEE Military Communications Conf.*, *Boston, 13–16 Oct. 2003.*
16. P. Sehier, P. Brelivet, and I. Aubry, "A New Jammer Estimation Method Adapted to FH-PSK Waveforms," *Proc. 1994 IEEE Military Communications Conf.* **3,** *2–5 Oct. 1994, Long Branch, N.J.,* pp. 1027–1031.
17. H. El Gamal and E. Geraniotis, "Iterative Channel Estimation and Decoding for Convolutionally Coded Anti-Jam FH Signals," *IEEE Trans. Commun.* **50** (2), pp. 321–331.
18. W.G. Phoel, "Serially Concatenated Convolutional Coding for Frequency-Hopped PSK in Partial-Band Jamming," *Proc. 2002 IEEE Military Communications Conf.* **2,** *Anaheim, Calif., 7–10 Oct. 2002,* pp. 763–767.
19. J. Hagenauer, E. Offer, and L. Papke, "Iterative Decoding of Binary Block and Convolutional Codes," *IEEE Trans. Inf. Theory* **42** (2), 1996, pp. 429–445.
20. P. Robertson, E. Villebrun, and P. Hoeher, "A Comparison of Optimal and Sub-Optimal MAP Decoding Algorithms Operating in the Log Domain," *Proc. 1995 IEEE International Conf. on Communications (ICC '95), Seattle, 18–22 June 1995*, pp. 1009–1013.
21. S. Benedetto, D. Divsalar, G. Montorsi, and F. Pollara, "Serial Concatenation of Interleaved Codes: Performance Analysis, Design, and Iterative Decoding," *IEEE Trans. Inf. Theory* **44** (3), 1998, pp. 909–926.

APPENDIX: TURBO DECODING AND THE APP ALGORITHM

The *a posteriori* probability (APP) decoding algorithm is based on the symbol-by-symbol maximum *a posteriori* (MAP) algorithm [1]. The MAP decoder makes each bit decision $\hat{b}_k$ to maximize the probability that that bit was sent, given the entire received sequence

$$\hat{b}_k = \arg\max_{b_k \in \{+1,-1\}} \Pr(b_k | \mathbf{r}),$$

where b_k is the kth bit of the block of N information bits encoded in the transmitted code word, and $\mathbf{r}$ represents the received sequence corresponding to the code word. To ease notation later in this appendix, we assume that the binary digits 0 and 1 are already mapped to +1 and –1. Figure A shows a simplified block diagram relating bits to code words. For each bit, we can calculate the log-APP ratio

$$\lambda_k = \log\left[\frac{\Pr(b_k = +1 \,|\, \mathbf{r})}{\Pr(b_k = -1 \,|\, \mathbf{r})}\right],$$

and decide that +1 was sent if $\lambda_k > 0$, and decide that –1 was sent otherwise. The turbo decoder differs from the MAP decoder in the following way. Rather than making hard decisions on the bits and stopping, the turbo-decoding algorithm keeps the soft information in the log-APP ratio and passes it between constituent decoders. The soft output of one decoder is called extrinsic information and is used as *a priori* information in the other decoder.

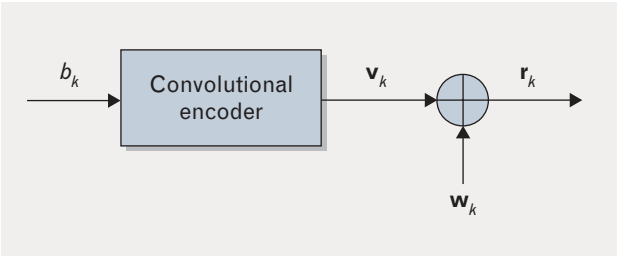

FIGURE A. A simplified block diagram showing the relationship between the input bits b_k, the code word subsequences $\mathbf{v}_k$ (where we assume that n code bits are generated for each input bit, so that $\mathbf{v}_k$ is an n-element column vector), and the noisy estimate of the code word subsequence $\mathbf{r}_k$. The vector $\mathbf{w}_k$ is an n-element column vector of white Gaussian noise samples.

The remainder of this appendix describes how to compute λ_k based on the code word estimate and *a priori* information. A detailed description of the APP algorithm and turbo decoding for PCCCs can be found elsewhere [2].

We calculate the APP by applying Bayes' rule,

$$\Pr(b_k | \mathbf{r}) = \frac{\Pr(b_k, \mathbf{r})}{\Pr(\mathbf{r})},$$

and note that $\Pr(\mathbf{r})$ is the same in the numerator and denominator of the APP ratio and therefore will cancel when computing λ_k. We then take advantage of the fact that the convolutional codes we are considering can be represented as finite state machines. The joint probability of b_k and $\mathbf{r}$, or $\Pr(b_k, \mathbf{r})$, can be computed by summing over the joint probabilities of $\mathbf{r}$ with all information sequences with the desired value for b_k. Because of the finite state machine representation, this expression can be written as the sum over all state transitions corresponding to the desired value of b_k

$$\begin{aligned}\Pr(b_k, \mathbf{r}) &= \sum_{\mathbf{b}|b_k} \Pr(\mathbf{b}, \mathbf{r}) \\ &= \sum_{(s',s)|b_k} \Pr(s_{k-1}=s', s_k = s, \mathbf{r}),\end{aligned}$$

where $\mathbf{b}$ is the sequence of N information bits, $\mathbf{b}|b_k$ denotes all information sequences with the desired bit value in the kth position, and s_k is the state of the convolutional code at index k.

The convolutional code can also be described on a trellis. Figure B shows the trellis for the outer convolutional code considered in this article. Each branch is labeled with the sign of the information bit that leads to that state transition, followed by the signs of the resulting code bits. For the discussion here, we can think of

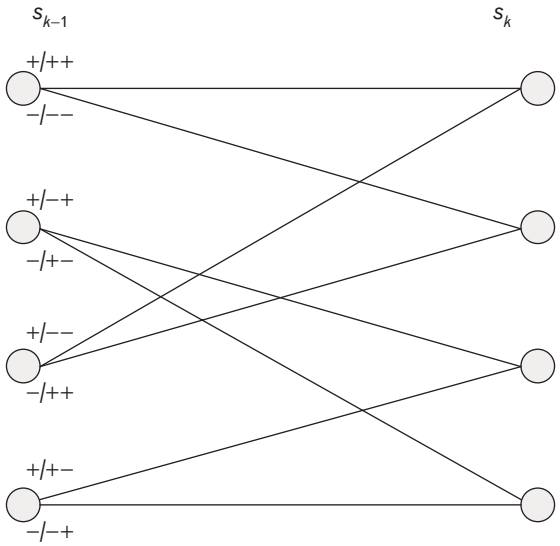

FIGURE B. Trellis diagram of outer convolutional code. There are four states at index $k-1$ and four at index k. Each state corresponds to distinct contents of the memory elements shown in the encoder in Figure 3. Each state at index $k-1$ is connected to only two of the four states of index k. Each branch is labeled with the sign of the bit corresponding to that state transition, followed by the signs of the resulting two code bits. The code word formed by encoding an N information bit block can be represented by a unique path through N consecutive stages of this trellis.

the code word as a path through N consecutive stages of the trellis. Note that for this rate-1/2 binary convolutional code, each stage of the trellis corresponds to one information bit and two code bits.

We now have a representation of the probabilities to be computed, based on the trellis structure of the convolutional code. It remains to derive $\Pr(s_{k-1}, s_k, \mathbf{r})$, the joint probability of $\mathbf{r}$ and the states at $k-1$ and k. We can separate the desired joint probability into three basic terms—$\alpha_{k-1}(s)$, $\beta_k(s)$, and $\gamma_k(s', s)$—as follows:

$$\begin{aligned}&\Pr(s_{k-1}=s', s_k=s, \mathbf{r})\\ &= \Pr(s_{k-1}=s', \mathbf{r}_1^{k-1}) \Pr(s_k=s, \mathbf{r}_k \,|\, s_{k-1}=s') \Pr(\mathbf{r}_{k+1}^{N} \,|\, s_k=s)\\ &= \alpha_{k-1}(s') \gamma_k(s', s) \beta_k(s),\end{aligned}$$

where the first term,

$$\begin{aligned}\alpha_{k-1}(s) &= \Pr(s_k=s, \mathbf{r}_1^{k-1})\\ &= \sum_{s'} \alpha_{k-2}(s') \gamma_{k-1}(s', s),\end{aligned}$$

is the joint probability of the sequence $\mathbf{r}$ through the $(k-1)$ stage and the state at $k-1$. The second term,

$$\begin{aligned}\gamma_k(s', s) &= \Pr(s_k=s, \mathbf{r}_k \,|\, s_{k-1}=s')\\ &= \Pr(s_k=s \,|\, s_{k-1}=s') \Pr(\mathbf{r}_k \,|\, s_k=s, s_{k-1}=s')\\ &= \Pr(b_k) \Pr(\mathbf{r}_k \,|\, b_k),\end{aligned}$$

is the joint probability of the code word subsequence at stage k, given the state at the previous stage. The third term,

$$\begin{aligned}\beta_k(s) &= \Pr(\mathbf{r}_{k+1}^{N} \,|\, s_k=s)\\ &= \sum_{s} \beta_{k+1}(s) \gamma_{k+1}(s', s),\end{aligned}$$

is the probability of the sequence $\mathbf{r}$ from stage $k+1$ through the end of the block, given the state at stage k. In these expressions, $\mathbf{r}_m^n$ represents the estimated code word subsequence from stage m through stage n, i.e., $\{\mathbf{r}_m, \mathbf{r}_{m+1}, \ldots, \mathbf{r}_n\}$. Note that both $\alpha_k(s)$ and $\beta_k(s)$ can be computed recursively. The computation of $\alpha_k(s)$ is called the forward recursion because $\alpha_k(s)$ is computed from $\alpha_{k-1}(s)$. Similarly, the computation of $\beta_k(s)$ is called the backward recursion because it is computed from $\beta_{k+1}(s)$. Furthermore, since each of these terms sums over the valid state transitions, we can use the trellis structure for visualizing these recursions; each node in the trellis is associated with an $\alpha_k(s)$ and a $\beta_k(s)$. The key to all of these is $\gamma_k(s', s)$, the parameter that contains the channel and *a priori* information and that determines the metrics associated with state transitions.

The term $\Pr(b_k)$ in calculating $\gamma_k(s', s)$ is the *a priori* probability. In the first iteration of the decoder, $\Pr(b_k)$ is 1/2 for both $b_k = +1$ and $b_k = -1$. In later iterations, the probability used comes from the other constituent decoder. Since the APP decoder produces log-APP ratios for the code bits, we need to express $\Pr(b_k)$ in terms of its log-probability ratio. Let λ'_k denote the logarithm of the *a priori* probability ratio for bit k. Then we can write

$$\Pr(b_k) = B_k \exp(b_k \lambda'_k / 2),$$

where B_k is a function of λ'_k but independent of b_k [2]. The second term in the calculation of $\gamma_k(s', s)$, $\Pr(\mathbf{r}_k|b_k)$, is the likelihood of receiving $\mathbf{r}_k$ given that b_k was sent. This probability is a function only of the channel values

for stage k,

$$\Pr(\mathbf{r}_k \mid b_k) = \left(\frac{1}{\sqrt{2\pi\sigma^2}}\right)^n \exp\left(\frac{-\|\mathbf{r}_k - \mathbf{v}_k\|^2}{2\sigma^2}\right)$$
$$= C_k \exp\left(\frac{1}{\sigma^2}\mathbf{r}_k^T \mathbf{v}_k\right),$$

where we assume that $\mathbf{r}_k = \mathbf{v}_k + \mathbf{w}_k$, as illustrated in Figure A, and $\mathbf{w}_k$ is a length-n vector of Gaussian random variables with variance σ^2 per element. It follows that we can write

$$\gamma_k(s', s) = B_k C_k \exp\left(\frac{1}{2} b_k \lambda'_k + \frac{1}{\sigma^2}\mathbf{r}_k^T \mathbf{v}_k\right).$$

We now have expressions to calculate the log-APP ratio from the code word estimate $\mathbf{r}$ and the log *a priori* probability ratio λ'_k, but performing these operations in the probability domain can be numerically unstable. It also requires many multiplications which, in general, are more complicated to implement than additions. Consequently, we consider how to implement the APP algorithm in the log domain. To do so, we first define the function

$$\max^*(a, b) \equiv \ln(e^a + e^b)$$
$$= \max(a, b) \ln(1 + e^{-|a-b|}),$$

which can be implemented by using a look-up table for the log term, and then only for values of $|a - b|$ small enough to make a significant difference. A variation on this decoder neglects the table look-up and computes the maximum only in each case where the max* operation occurs. Such a decoder is often referred to as the max-log-APP or max-log-MAP decoder. The max-log-APP decoder tends to perform within a fraction of a decibel of the log-APP decoder but is simpler to implement.

We can now update each of $\alpha_k(s)$, $\beta_k(s)$, and $\gamma_k(s', s)$ terms as follows:

$$\ln[\alpha_k(s)] = \max_{s'}{}^* \left\{\ln[\alpha_{k-1}(s')] + \ln[\gamma_k(s', s)]\right\},$$

$$\ln[\beta_k(s')] = \max_{s}{}^* \left\{\ln[\beta_{k+1}(s)] + \ln[\gamma_{k+1}(s', s)]\right\},$$

and

$$\ln[\gamma_k(s', s)] \approx \frac{1}{2}\left(b_k \lambda'_k + \frac{2}{\sigma^2}\mathbf{r}_k^T \mathbf{v}_k\right),$$

where the term due to B_k and C_k is omitted because it will cancel in later steps. Consequently, the log-APP ratio is given by

$$\lambda_k = \max_{(s',s)|b_k=+1}{}^* \left\{\ln[\Pr(s_{k-1}=s' \mid s_k=s, \mathbf{r})]\right\}$$
$$- \max_{(s',s)|b_k=-1}{}^* \left\{\ln[\Pr(s_{k-1}=s' \mid s_k=s, \mathbf{r})]\right\}$$
$$= \max_{(s',s)|b_k=+1}{}^* \left\{\ln[\alpha_{k-1}(s')] + \ln[\gamma_k(s', s)] + \ln[\beta_k(s)]\right\}$$
$$- \max_{(s',s)|b_k=-1}{}^* \left\{\ln[\alpha_{k-1}(s')] + \ln[\gamma_k(s', s)] + \ln[\beta_k(s)]\right\}.$$

In general, the values for $\alpha_k(s)$ are computed by starting with $k = 1$ up through $k = N$, and the $\beta_k(s)$ are computed by beginning with $k = N$ and working backward to $k = 1$.

The extrinsic information used as *a priori* information in another constituent decoder is formed differently, depending on the configuration of the concatenated code. For example, with a PCCC using systematic constituent codes (i.e., the information bits are contained unaltered within the code word), the extrinsic information passed from one decoder to another is a log-probability ratio of the information bits. However, for the iterative algorithm to remain stable, we need to be careful not to inadvertently reuse information in the same decoder. Specifically, for a particular constituent decoder, inputs directly related to the information bit probabilities are the channel values for the systematic bits and the *a priori* information from the other decoder. Because the systematic channel values will also be input to the other constituent decoder, we need to remove that information from the extrinsic information passed between decoders so that it is not counted twice. In addition, since the *a priori* information in one decoder came from the other decoder, that information too should be removed from the log-APP ratio before it is passed back as extrinsic information. Therefore, for the typical PCCC, the value output as extrinsic information on bit k is given by $\lambda_k - v_k^s - \lambda'_k$, where v_k^s denotes the channel value corresponding to the systematic bit k.

For the SCCC configuration considered in this article, the situation is somewhat different. First, neither of

the codes used is systematic. Second, the actual channel values input to the inner decoder are not directly input to the outer constituent decoder. Let's consider the inner decoder first. The inputs to this decoder are the channel values and the *a priori* information from the outer decoder. In this case, what are considered the "information" bits of the inner code are actually the code bits of the outer code. The inner decoder implements the APP algorithm and computes the log-probability ratio λ_k for the outer code bits. As discussed above, before this information can be passed to the outer decoder, the *a priori* information needs to be removed. Therefore, the inner decoder passes to the outer decoder $\lambda_k - \lambda'_k$, which the outer decoder interprets as the channel values or, equivalently, the code bit estimates. The outer decoder then performs the APP algorithm on these received values. However, with this serial concatenation, the log-APP ratio computed is used only for the hard decisions at the end of the iterations. It is extrinsic information on the *code bits* that needs to be computed and passed to the inner decoder. This is done in the same way as for the information bits, except that the max* operation is performed over all state transitions with the same value of the particular code bit being considered.

REFERENCES

1. L.R. Bahl, J. Cocke, F. Jelinek, and J. Raviv, "Optimal Decoding of Linear Codes for Minimizing Symbol Error Rate," *IEEE Trans. Inf. Theory* **20**, Mar. 1974, pp. 284–287.
2. W. Ryan, "Concatenated Convolutional Codes and Iterative Decoding," in *Wiley Encyclopedia of Telecommunications, Vol. 1*, J.G. Proakis, ed. (Wiley, New York, 2003), pp. 556–570.

WAYNE G. PHOEL
is a staff member in the Advanced Satcom Systems and Operations group. His current research interests are in the areas of communications and information theory with emphasis on modulation and coding. At Lincoln Laboratory he has worked primarily on the design of the protected radio frequency waveform for the Transformational Satellite Communications System (TSAT), as well as modulation and error-control coding for payload and terminal emulators for the Advanced EHF system. Prior to joining the Laboratory, he contributed to third-generation terrestrial cellular communications research at Motorola Cellular Infrastructure Group, in Arlington Heights, Illinois, and also worked on signal processing for radar applications at Calspan Corporation, in Buffalo, New York. He has B.S. and M.Eng. degrees in electrical engineering from Cornell University, and a Ph.D. degree in electrical and computer engineering from Northwestern University.