

# Low complexity serially concatenated coding for the deep space optical channel

Bruce Moision, Jon Hamkins\*  
Jet Propulsion Laboratory, California Institute of Technology  
4800 Oak Grove Dr., Pasadena, CA 91109-8099  
Email: {bruce.moision, jon.hamkins}@jpl.nasa.gov

October 31, 2002

## Abstract

We illustrate the performance of low-complexity serially concatenated codes designed for the deep space optical channel. The codes are an iteratively decoded, serial concatenation of a convolutional code with coded  $M$ -ary pulse-position-modulation (PPM) through a bit interleaver. For  $M = 256$  with a 4096-bit interleaver, we illustrate performance 1.2dB from capacity and gains of 2.8dB relative to a baseline (255, 127) Reed-Solomon code concatenated with PPM. Storing channel likelihoods may be prohibitively expensive for iterative decoding of high PPM orders. We show that the receiver may compute and store a small subset of the channel likelihoods and suffer negligible performance degradation. The complexity of the soft decision decoding is also reduced. For  $M = 256$  we show negligible performance degradation when only 8 of each 256 likelihoods are stored. In this case, the number of operations for the forward-backward algorithm on the inner code, which comprises the bulk of operations, may be reduced by 32%.

## 1 Introduction

NASA is developing optical links to support deep space communication at data rates on the order of 100 Mbit/second. These optical links operate efficiently at high peak to average power ratios which may be achieved by modulating the data using  $M$ -ary pulse-position-modulation (PPM). For certain lasers and detectors, the optimal PPM order is high— $M \geq 256$ . High PPM orders and data rates require short pulse widths. In fact the optical pulse widths may outstrip the speed of the digital hardware required for implementing many candidate encoders and decoders. Hence, it is important that the encoder and decoder are low-complexity. We present a low-complexity iteratively decoded convolutionally coded modulation that significantly outperforms baseline Reed-Solomon (RS) coded PPM.

High PPM orders also imply low code rates. Storing the channel likelihoods, which are required for iterative decoding, can be prohibitively expensive for large interleavers and PPM orders. We show that the storage requirements can be reduced by storing a subset of the likelihoods while suffering no loss in performance. The complexity of implementing the forward-backward algorithm is also reduced when partial likelihoods are retained.

## 2 Serially concatenated coding

Our discrete-time coded binary communications channel model is illustrated in Fig. 1. User data is encoded by the serial concatenation of (outer code)  $\mathcal{C}_o$  and (inner code)  $\mathcal{C}_i$  through a bit interleaver  $\Pi$ . Each transmitted codeword, a binary

---

\*The work described was funded by the IPN Technology Program and the Caltech Presidents Fund and performed at the Jet Propulsion Laboratory, California Institute of Technology under contract with the National Aeronautics and Space Administration.

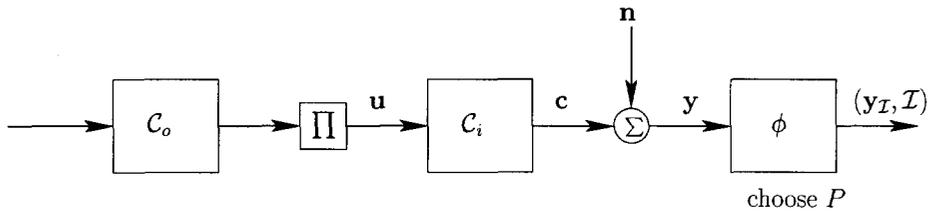


Figure 1: Constrained storage channel model

$M$ -vector  $\mathbf{c}$ , has noise  $\mathbf{n}$  added such that the receiver observes the noisy version  $\mathbf{y} = \mathbf{c} + \mathbf{n}$ . We add the constraint that only  $P$  of each  $M$  observations, as well as their indices  $\mathcal{I}$ , are made available to the receiver. The mapping  $\phi$  denotes the rule for choosing the  $P$  samples,  $\phi : \mathbf{y} \rightarrow (\mathbf{y}_{\mathcal{I}}, \mathcal{I})$ ,  $\mathcal{I} \subset \{1, \dots, M\}$ ,  $|\mathcal{I}| = P$ , and  $\mathbf{y}_{\mathcal{I}}$  is the vector of retained samples  $y_i, i \in \mathcal{I}$ . A set of indices used as a subscript to a vector denotes the vector formed from the indexed components.

We use the notation  $C_o \rightarrow C_i$  to denote the non-iteratively decoded serial concatenation of outer code  $C_o$  and inner code  $C_i$  and  $C_o \leftrightarrow C_i$  to denote iterative decoding. We address PPM order  $M = 256$  and make comparisons with a (255, 127) Reed-Solomon code. Prior work investigated the system PCCC  $\rightarrow$  PPM [1, 2] for the optical channel, where PCCC is an iteratively decoded parallel concatenated convolutional code. Peleg and Shamai [3] investigated the system PCCC  $\leftrightarrow$  PPM on a discrete-time Rayleigh-fading model, illustrating performance 1–2 dB from capacity. The architecture we consider is simpler than that in [1, 2, 3].

We consider a number of systematic, rate 1/2 convolutional codes for  $C_o$ . The following table lists the numerators and denominators in octal notation of the codes considered.

Code	numerator	denominator	states	$d_{min}$
cc1	1	1	1	2
cc2	4	6	2	3
cc3	5	7	4	5
cc4	74	64	8	6

Table 1: Outer convolutional codes

We use an AWGN model, which is a reasonable model for an avalanche photo-diode (APD) detector when the number of incident signal photons is large. In AWGN,  $\mathbf{n}$  is a vector of independent, identically distributed zero-mean Gaussian random variables with variance  $\sigma_n^2 = N_0/2$ . The PPM mapping is preceded by a binary accumulator, cc2 in Table 1, making the inner code recursive. We refer to the inner code that is formed by the concatenation of a binary accumulator and PPM mapping as accumulate-PPM (APPM).

Figure 2 illustrates the performance for  $C_o \leftrightarrow$  APPM with  $P = M$  (no storage constraint) as a function of the bit-SNR,  $E_b/N_0$ . The composite code rate is  $R = \log_2 M/2M = 1/64$ . Performance is compared to the capacity for rate  $R$  constrained to use a  $M$ -PPM alphabet and a (255, 127) RS code. All cases use a 4096-bit spread interleaver. A stopping rule that terminates when the inner code produces a codeword of the outer code is used for all but the cc1 code, whose low distance renders this stopping rule useless. A maximum of 64 iterations are allowed, and fewer than 7 iterations are typical at bit error rates below  $10^{-5}$ . At a bit error rate of  $10^{-5}$ , the best serially concatenated code is 1.2dB from capacity and gains 2.8dB over the Reed Solomon baseline. Additional gains of .2–.5 dB are achievable by increasing the interleaver size to 65536 bits.

### 3 Partial Statistics

To realize the gains of the iterative decoding algorithms requires likelihoods to be computed and stored for each codeword of the inner code. For an inner code that maps to  $M$ -ary PPM symbols, the storage required at the decoder for one frame of channel likelihoods is  $fM|\Pi|/\log_2(M)$  bits, where  $f$  is the number of bits used to represent fixed or floating-point values. High data rates, large values of  $M$  and large interleavers can make likelihood computation and storage prohibitively expensive. To reduce the complexity of iterative decoding, we take  $P < M$  and compute and store only a subset of the channel likelihoods.

The conditional likelihoods

$$p(\phi(\mathbf{y})|\mathbf{c}) = p(\mathbf{y}_{\mathcal{I}} = \mathbf{y}_{\mathcal{I}}|\mathbf{c} = c)p(\mathcal{I} = \mathcal{I}|\mathbf{c} = c, \mathbf{y}_{\mathcal{I}} = \mathbf{y}_{\mathcal{I}}), \quad (1)$$

are a sufficient statistic for the maximum-a-posteriori estimation of  $\mathbf{u}$  given  $\phi(\mathbf{y})$ , and serve as input likelihoods for iterative decoding. The term  $p(\mathbf{y}_{\mathcal{I}}|\mathbf{c})$  is the likelihood that would result if the input were mapped to a  $P$  dimensional constellation and the term  $p(\mathcal{I}|\mathbf{c}, \mathbf{y}_{\mathcal{I}})$  is an adjustment to reflect the outcome of the decision.

For moderate  $M$ , PPM is a sparse on-off-keying, hence a reasonable choice is to let  $\phi$  choose the  $P$  largest elements of  $\mathbf{y}$ . Then

$$\begin{aligned} p(\mathcal{I}|\mathbf{c}, \mathbf{y}_{\mathcal{I}}) &= p(\max_{\bar{\mathcal{I}}} y_{\bar{\mathcal{I}}} \leq \min_{\mathcal{I}} y_{\mathcal{I}}|\mathbf{c} = c) \\ &= K \begin{cases} g(\psi) & , |c_{\mathcal{I}}| = 1 \\ 1 & , |c_{\mathcal{I}}| = 0 \end{cases}, \end{aligned} \quad (2)$$

where the maximum of a vector is the largest element of the vector,  $\bar{\mathcal{I}} = \{1, \dots, M\}/\mathcal{I}$ ,  $\psi = \min_{\mathcal{I}} y_{\mathcal{I}}$ ,  $K$  is a constant and

$$g(\psi) = \frac{F_n(\psi)}{F_n(\psi - 1)},$$

where  $F_n$  is the cumulative density function of a noise sample. Computing likelihoods via (2) requires a table lookup or computation in order to determine  $g(\psi)$ . This can be eliminated by replacing  $g(\psi)$  with an estimate independent of  $\psi$ . We have observed negligible degradation when  $g(\psi)$  is replaced with its mean. Substituting  $p(\mathbf{y}_{\mathcal{I}}|\mathbf{c})$  for the AWGN channel we have

$$p(\phi(\mathbf{y})|\mathbf{c}) \approx K \begin{cases} E(g(\psi)) \exp(y_j/\sigma^2) & , |c_{\mathcal{I}}| = 1 \\ 1 & , |c_{\mathcal{I}}| = 0, \end{cases} \quad (3)$$

where  $j$  is the element of  $\mathcal{I}$  such that  $c_j = 1$ . Figure 3 illustrates performance with  $P \in \{1, 2, 4, 8, M\}$  for cc3 $\leftrightarrow$ APPM. All cases use 8 iterations and a 4096-bit spread interleaver. We see 0.1 dB degradation when 1/64 of the likelihoods are kept, and negligible degradation when 1/32 are kept.

#### 3.1 Complexity, channel likelihoods

On observation of  $\mathbf{y}_k$ , the  $P$  largest elements of  $\mathbf{y}_k$  are determined, their corresponding likelihoods are computed and stored. Table 2 lists the number of operations and storage required to complete this operation with full and partial statistics. To assess sorting cost, we assume the first  $P$  observations are sorted by insertion and the following  $M-P$  by a heapsort [4, pp.344]. In addition to storing the floating point values, we must save the addresses of the  $P$  largest values, which we assess a cost of  $P \log P$  bits.

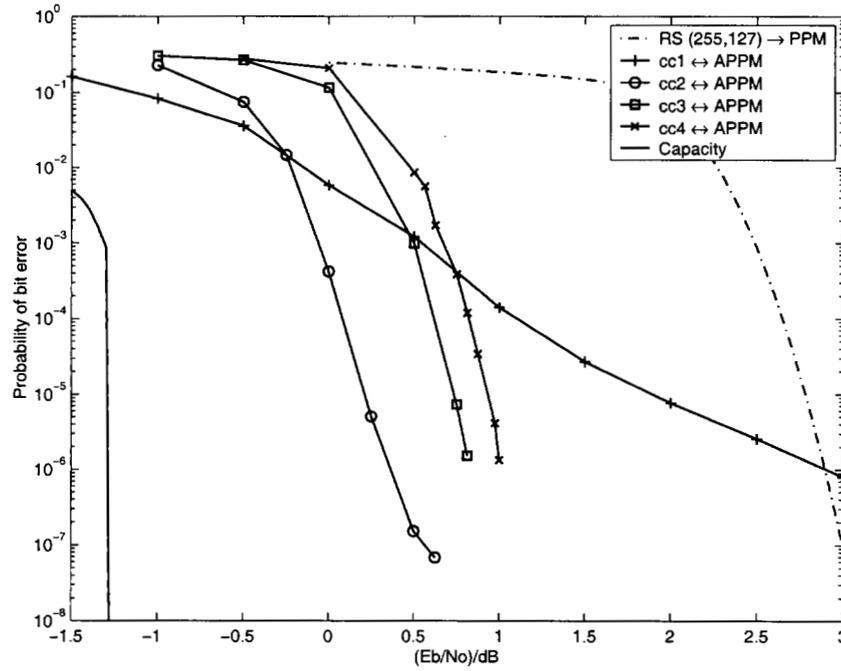


Figure 2: Performance with full statistics,  $M = 256$ ,  $C_o \leftrightarrow$  APPM with  $|\Pi| = 4096$

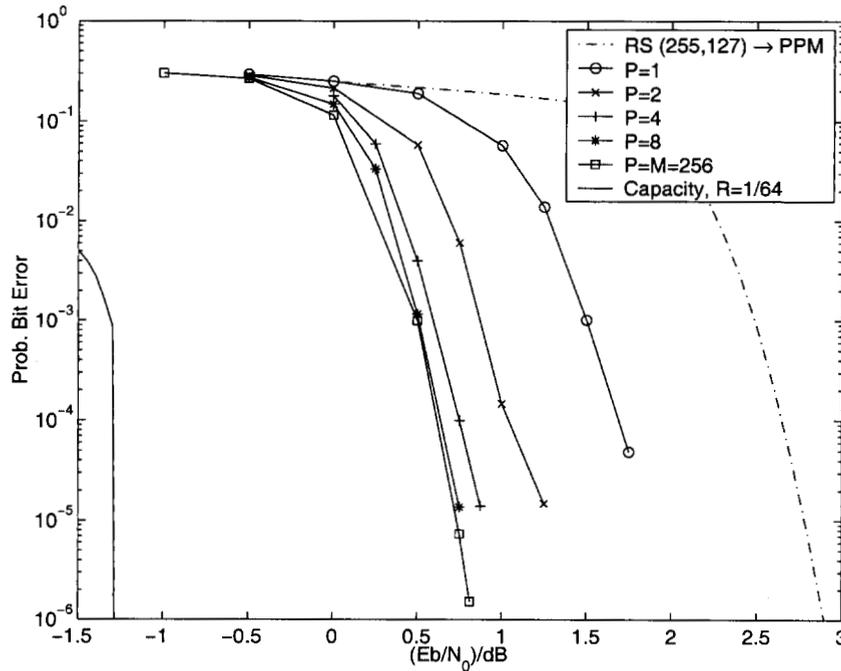


Figure 3: Performance with partial statistics,  $M = 256$ ,  $P \in \{1, 2, 4, 8\}$ ,  $cc3 \leftrightarrow$  APPM with  $|\Pi| = 4096$ , 8 iterations

	Operations			Storage(bits)	
	mult.	exp.	comparisons in sort	addresses	likelihoods
full	$M$	$M$	–	–	$Mf$
partial	$P$	$P$	$(P-1)^2 + (M-P)\log P$	$P\log P$	$Pf$

Table 2: Storage and computation cost to compute channel likelihoods

For our case of interest, the code is APPM, such that  $|\mathcal{V}| = 2$  and  $|\mathcal{E}| = 2M$ . With  $M = 256$ ,  $P = 8$  and  $f = 4$ , partial statistics require  $\approx 1/18$  times the storage as full. The computational complexity depends on the platform, since we are trading off comparisons (additions) for multiplications and exponentiations.

## 4 Complexity, forward-backward algorithm

The inner code maps a block of information symbols  $\mathbf{u} = (\mathbf{u}_1, \dots, \mathbf{u}_N)$  to codeword  $\mathbf{c} = (\mathbf{c}_1, \dots, \mathbf{c}_N)$ , where  $\mathbf{u}_k, \mathbf{c}_k$  are binary vectors with  $i$ th component  $\mathbf{u}_{k,i}, \mathbf{c}_{k,i}$ . The code is described by a time-invariant graph (straight-forward modifications would treat time-varying graphs) consisting of a set of states  $\mathcal{V}$ , and a set of directed, labeled edges  $\mathcal{E}$ . Each edge  $e \in \mathcal{E}$  has an initial state  $i(e)$ , a terminal state  $t(e)$ , an input label  $u(e)$  and output label  $c(e)$ . We assume that encoding proceeds by following a path through the graph and reading off the output edge labels as follows. Let  $\mathbf{s}_{k-1}$  be the state at time  $k-1$ , and  $\mathbf{e}_k$  the edge with  $i(\mathbf{e}_k) = \mathbf{s}_{k-1}$  and  $\mathbf{u}_k = u(\mathbf{e}_k)$ . Then  $\mathbf{c}_k = c(\mathbf{e}_k)$  and  $\mathbf{s}_k = t(\mathbf{e}_k)$ . Throughout we use shorthand  $p(\mathbf{u}_k) = p(\mathbf{u}_k = u)$  when the realization is clear from context.

Each iteration, the forward-backward algorithm [5] begins by computing

$$\gamma_k(e) = p(\mathbf{u}_k = u(e))p(\phi(\mathbf{y}_k)|\mathbf{c} = c(e)),$$

for each edge in the trellis, which changes each iteration as  $p(\mathbf{u}_k)$  is updated by the outer code. However, with partial statistics, there are only  $P+1$  distinct values of  $p(\phi(\mathbf{y}_k)|\mathbf{c}_k)$ . One can take advantage of this and use a reduced complexity time-varying trellis with  $|\mathcal{V}|$  states and at most  $|\mathcal{V}|(P+|\mathcal{V}|)$  edges, reducing the computational complexity in our cases of interest.

Let  $\mathcal{J}_k(r, s, \xi)$  be the collection of parallel edges in the  $k$ -th trellis stage with channel likelihood  $\xi$ ,

$$\mathcal{J}_k(r, s, \xi) = \{e | i(e) = r, t(e) = s, p(\phi(\mathbf{y}_k)|\mathbf{c} = c(e)) = \xi\}.$$

Form a *partial* trellis by replacing the edges in  $\mathcal{J}_k(r, s, \xi)$  with a single edge  $e(r, s, \xi)$  with initial state  $r$  and terminal state  $s$ . Let  $\mathcal{E}'_k$  be the collection of modified edges. The partial trellis will have  $|\mathcal{E}'_k| \leq |\mathcal{V}|(P+|\mathcal{V}|)$ . For  $e \in \mathcal{E}'_k$  we put

$$\gamma_k(e) = p(\mathbf{u}_k(e) \in \mathcal{J}_k(i(e), t(e), p(\phi(\mathbf{y}_k)|\mathbf{c} = c(e))))p(\phi(\mathbf{y}_k)|\mathbf{c} = c(e)).$$

We proceed to compute  $\lambda_k(e) = p(\mathbf{e}_k = e, \mathbf{y})$  for each edge in the partial trellis using the forward-backward algorithm. Note that for  $e \in \mathcal{J}_k(r, s, \xi) \subset \mathcal{E}$ ,

$$\lambda_k(e) = \lambda_k((e(r, s, \xi)) \frac{p(\mathbf{u}_k = u(e))}{p(\mathbf{u}_k(e) \in \mathcal{J}_k(r, s, p(\phi(\mathbf{y}_k)|\mathbf{c} = c(e))))} \quad (4)$$

Hence, after computing the  $\lambda$ 's on the partial trellis we may compute the bit likelihoods  $p(\mathbf{u}_{k,i}|\mathbf{y})$  as

$$p(\mathbf{u}_{k,i} = 0|\mathbf{y}) = \frac{1}{p(\mathbf{y})} \sum_{e:e \in \mathcal{E}, u_i(e)=0} \lambda_k(e)$$

$$= \frac{1}{p(\mathbf{y})} \sum_{e:e \in \mathcal{E}'_k} \frac{\lambda_k(e)}{p(\mathbf{u}_k(e) \in \mathcal{J}_k(i(e), t(e), p(\phi(\mathbf{y}_k)|\mathbf{c} = c(e)) = \xi))}$$
 (5)

$$\sum_{e' \in \mathcal{J}_k(i(e), t(e), \xi), u_i(e')=0} p(\mathbf{u}_k = u(e))$$
 (6)

For the inner code APPM with  $M = 256, P = 8$  we require 32% fewer operations on the partial trellis relative to the full trellis. Storage requirements per trellis stage are also reduced, but we presume the algorithm is implemented with a sliding-window such that the storage costs are dominated by the channel likelihoods.

## 5 Conclusions

We have illustrated performance within 1.5dB of capacity is achievable via iterative decoding of a simple serial concatenation of a convolutional code with accumulate-PPM through a relatively short bit interleaver. For channels that modulate to a high PPM order, the storage required for the channel likelihoods may be a bottleneck in implementing iterative decoding. We have shown that a small subset of the channel likelihoods may be used with negligible degradation, by setting the remainder of the likelihoods to an appropriate constant. In addition, a reduced complexity trellis may be used for the forward-backward algorithm, reducing the number of operations required.

## References

- [1] J. Hamkins, "Performance of binary turbo coded 256-PPM," *TMO Progress Report*, vol. 42, pp. 1–15, Aug. 1999.
- [2] K. Kiasaleh, "Turbo-coded optical PPM communication systems," *Journal of Lightwave Technology*, vol. 16, pp. 18–26, Jan. 1998.
- [3] M. Peleg and S. Shamai, "Efficient communication over the discrete-time memoryless rayleigh fading channel with turbo coding/decoding," *European Transactions on Telecommunications*, vol. 11, pp. 475–485, September-October 200.
- [4] W. H. Press, S. A. Teukolsky, W. T. Vetterling, and B. P. Flannery, *Numerical Recipes in C*. New York: Cambridge University Press, 2 ed., 1992.
- [5] L. Bahl, J. Cocke, F. Jelinek, and J. Raviv, "Optimal decoding of linear codes for minimizing symbol error rate," *IEEE Transactions on Information Theory*, vol. 20, pp. 284–287, Mar. 1974.