Contents

Constrained 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

May 20, 2002

Abstract

We investigate methods of coding for a channel subject to a large dead-time constraint, i.e., a constraint on the minimum spacing between transmitted pulses, with the deep-space optical channel as the motivating example. Several constrained codes designed to satisfy the dead-time constraint are considered and compared on the basis of throughput, complexity, and decoded error-rate. The performance of an iteratively decoded serial concatenation of a constrained code with an outer code is evaluated and shown to provide significant gains over Reed-Solomon concatenated with Pulse Position Modulation.

1 Introduction

A free-space optical communications system is most efficient when the peak to average power ratio of the signal is large [1, 2]. These large ratios can be achieved by M-ary Pulse Position Modulation (PPM), in which $\log_2 M$ bits choose the location of a single pulsed slot in an M-slot frame. In theory, PPM can lead to an unbounded capacity [3], but in practice bandwidth constraints place a limit on capacity [4]. Nevertheless, when no noise is present in the system, it has been shown that PPM is near capacity-achieving [1, 5].

A Q-switched laser works well with the PPM format [6, 7], because it can successfully confine a large pulse energy to a narrow slot. One side effect of Q-switched lasers, however, is a required delay, or dead-time, between pulses during which the laser is recharged. This delay is significant relative to the pulse duration. PPM may be modified to satisfy the dead-time constraint by following each frame by a period during which no pulses are transmitted. However, this affects the optimality of PPM as a modulation format.

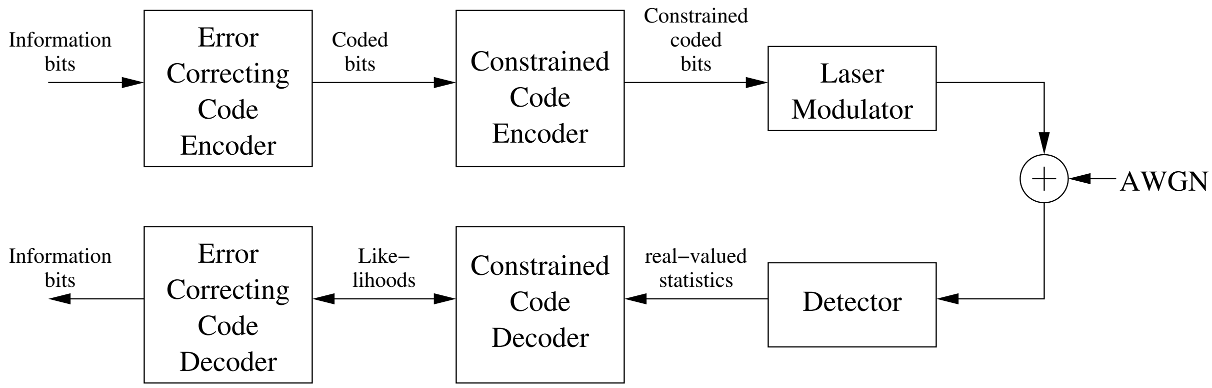

Figure 1: The communications system considered in this article.

There are more efficient–measured in throughput, or bits/second–ways to transmit information over a channel subject to a dead-time constraint. The problem of signaling efficiently under such a constraint has been well studied for applications in magnetic storage, where a similar restriction is imposed to compensate for the interference between magnetic media corresponding to closely recorded bits. Efficient signaling is affected by a modulation, or *constrained*, code. The deep-space problem is novel in that the dead-time is very large relative to the slot duration–on the order of 256-1024 times the slot duration compared to 1-2 times in magnetic storage applications.

This report investigates the application of constrained codes to the deep-space optical channel, and is organized as follows. In Section 2, we describe the channel model and introduce notation, in Section 3 we quantify the potential throughput gains available from constrained codes, in Section 4 we discuss implementation issues of various approaches, and in Section 5 we demonstrate the implementation of a constrained code in an iterative coding scheme.

2 Preliminaries

This article considers the binary-input, real-valued output channel model shown in Fig. 1. First, information bits are encoded using an error correcting code. Next, the constrained code takes these coded bits and encodes them further in a way that ensures the laser can physically transmit them, as we explain here. Time is partitioned into slots of duration T_s during which the laser may either transmit a pulse (a one) or not transmit a pulse (a zero), i.e., On-Off Keying (OOK)–see Fig. 2. In unconstrained OOK, a zero or one may appear in any position within the sequence of transmitted binary symbols. Q-switched lasers requiring dead-time T_d between pulses impose the constraint that at least $d \stackrel{\text{def}}{=} \lceil T_d/T_s \rceil$ zeros occur between ones. Slot synchronization, typically implemented by an early-late gate tracking

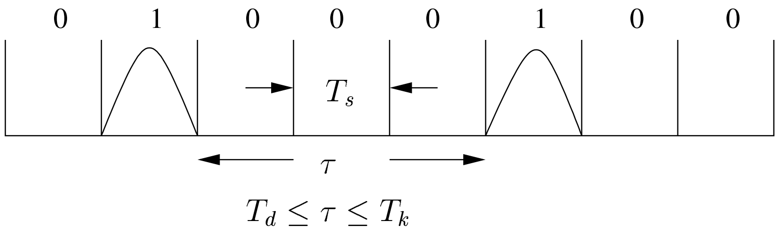

Figure 2: The optical channel is constrained on-off keying. A one represents a pulsed slot, and a zero a non-pulsed slot. There is at least T_d and at most T_k seconds between pulsed slots.

loop, imposes an additional constraint that no more than T_k seconds may elapse between pulses. In the transmitted binary sequence, this corresponds to the constraint that no more than $k \stackrel{\text{def}}{=} \lceil T_k/T_s \rceil$ zeros occur between ones. Together, the two constraints are referred to as a (d, k) *constraint* and the collection of sequences satisfying the constraint as a (d, k) *constrained system.* An invertible mapping of unconstrained binary sequences into the (d, k)-system is referred to as a *constrained code* [8,9]. Although it may be advantageous to violate the dead-time constraint, sending lower SNR pulses at shorter intervals, in this work it is assumed the dead-time T_d is a hard constraint, i.e., pulses must be separated by T_d seconds. The constrained code encoder in Fig. 1 makes sure the (d, k) constraint is satisfied.

At the receiver, light is focussed on a detector. The detector output can be either discrete or continuous, depending on the type of detector. For example, the output of a photon counting detector is the number of detected photons, which has a Poisson distribution. In most detectors— including photomultiplier tubes (PMT's), Avalanche Photo-Diode (APD) detectors, and even coherent detectors— the output is a real-valued voltage or current that arises from the detector input as well as from random processes within the detector and follow-on circuitry. These effects may be modeled in a variety of ways: the Poisson model is often used for PMT's, although a more accurate model is known in that case [10]; a Gaussian, Webb, or Webb plus Gaussian model can be used for APD's [11]; and a Gaussian model is appropriate for a coherent detector.

Throughout this paper, we shall use a Gaussian model for statistics called the AWGN-1 model [11], in which the slot statistics at the output of the detector are independent and of the form $y = s + n$, where $s \in \{0, 1\}$ is the binary symbol transmitted, and n is zero-mean Gaussian noise with variance $\sigma^2 = N_0/2$. The symbol energy is $E_s = E[s^2]$, so that, when used with a rate R_c bits/slot code,

$$\frac{E_b}{N_0} = \frac{E_s}{R_c N_0} = \frac{E[s^2]}{2R_c\sigma^2}.$$

We rely on the AWGN-1 channel model and report results as a function of the bit SNR, E_b/N_0, due to its simplicity, and the fact that all the above mentioned channel models behave in a way that is largely dominated by a bit SNR parameter [11] analogous to E_b/N_0. Hence, we expect coding results presented here will apply to a wide variety of channel models, in the sense that the relative coding gains of the various schemes will be about the same under different channel models and operating points.

Performance results in the literature are often given in terms of bits per photon, not E_b/N_0. This is a useful metric when the statistics are Poisson, particularly with low background light. However, stating results in bits per photon for the Gaussian channel model requires a determination of the relationship between E_b and the number of photons emitted from the laser. This is typically a non-linear relationship, and varies greatly from laser to laser. That is, results stated in terms of bits per photon would only apply to a particular choice of laser having that E_b to bits-per-photon relationship. Instead, by keeping the results in terms of E_b/N_0, we can make general conclusions and apply the conversion to bits per photon for a variety of potential lasers.

Throughout this article, we compare the new schemes against a baseline scheme in which, in Fig. 1, the error correcting code is a Reed-Solomon code, and the constrained code is PPM with added dead-time.

3 Throughput of various constrained codes

The k constraint is considered a design parameter–smaller values being preferred, unlike the d constraint, which must be satisfied. Hence, we will investigate achievable rates of cods into the (d, ∞) constraint–i.e. no maximum run-length constraint. Any constrained system may be described by a directed, labeled graph. Fig. 3 illustrates a graph presenting the (d, ∞) constrained system, where 0^x denotes a string of x zeros.

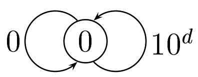

Figure 3: Graph presenting the (d, ∞) system.

The system is the set of sequences obtained by reading the labels of paths on the graph. The *capacity* of the (d, ∞) system when used on an error-free channel,

$$C(d) \stackrel{\text{def}}{=} \lim_{n \to \infty} \frac{1}{n} \log_2 |\text{words of length } n \text{ in the } (d, \infty) \text{ system}| \text{ bits/slot}, \tag{1}$$

is the asymptotic growth rate of the number of distinct words, i.e., finite-length patterns, in the system and the least upper bound on the rate of a code into the system. From [12], we have $C(d) = \ln(\lambda)$

d	Capacity (bits/s)	E_{PPM}	E_{TPPM}	E_{STPPM}	E_{NPM}
1	.6942	.5787	.8361	.7202	
2	.5515	.6057	.8524	.6045	.9067
4	.4057	.6382	.8701	.8217	
8	.2788	.6729	.8871	.7175	.8968
16	.1813	.7069	.9020	.7880	
32	.1130	.7380	.9146	.8847	
64	6.785×10^{-1}	.7702	.9304	.8670	.9158
128	4.008×10^{-2}	.7921	.9364	.9242	
256	2.319×10^{-2}	.8116	.9420	.9375	
512	1.320×10^{-2}	.8286	.9471	.9352	
1024	7.418×10^{-3}	.8434	.9516	.9427	
2048	4.124×10^{-3}	.8562	.9556	.9435	.9472

Table 1: Capacity of (d, ∞) constrained codes and relative efficiencies of some particular schemes, all to 4 significant digits

nats/slot, where λ is the largest positive root of

$$\lambda^{-(d+1)} + \lambda^{-1} - 1 = 0. \tag{2}$$

For small d exact solutions may be found efficiently for (2). For large d substitute $\lambda = e^{C(d)}$ and use the approximation $e^{-C(d)} \approx 1 - C(d)$ which yields $C(d)e^{(d+1)C(d)} \approx 1$ and thus

$$d + 1 \approx (d+1)C(d)e^{(d+1)C(d)},$$

or

$$C(d) \approx \frac{W(d+1)}{T_s(d+1)\ln 2} \text{ bits/s,} \tag{3}$$

where $W(z)$ is the *productlog* function which gives the solution for w in $z = we^w$. Table 1 lists capacities for $T_s = 1$.

With $R_{\mathcal{C}}(d)$ denoting the rate of a constrained code $\mathcal{C}$ into the (d, ∞) system, $E_{\mathcal{C}(d)} \stackrel{\text{def}}{=} R_{\mathcal{C}}(d)/C(d)$ is the relative *efficiency* of the code, measuring how close the code rate is to the limit. There are well-known techniques to construct codes into a constrained system at rates arbitrarily close to capacity, e.g., [8, 9]. However, for our parameter range, a straight-forward application of these approaches may be prohibitively complex. In the following sections we present some approaches that trade efficiency for complexity.

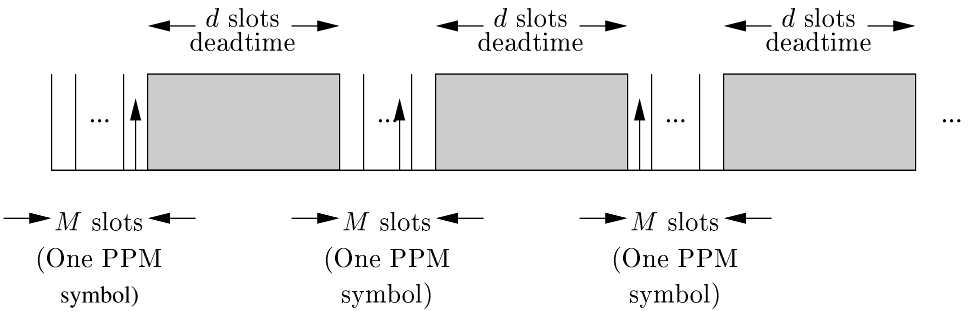

Figure 4: PPM signaling with a dead-time constraint.

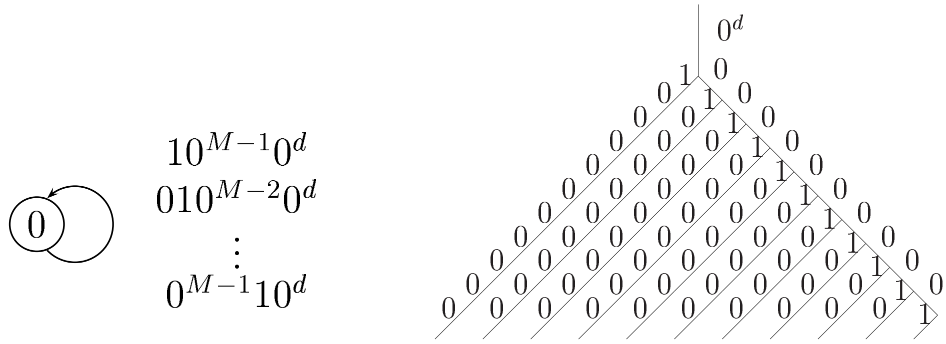

Figure 5: Graph and tree presenting PPM code sequences.

3.1 Pulse position modulation with added dead-time

First consider the efficiency of what will be considered the baseline, shown in Fig. 4. It consists of a constant M-slot PPM frame followed by a d-slot dead-time. A graph and tree presenting the allowable PPM code sequences with a dead-time constraint are illustrated in Fig. 5. Allowable sequences are read off the graph as described above. Code sequences on the tree are generated by traversing the tree. Considering PPM as a (d, ∞) constrained code, the rate is

$$R_{\text{PPM}}(d, M) = \frac{\log_2(M)}{T_s(M+d)} \text{ bits/s.}$$

For a given value of d we find M^*, the argument that maximizes $R_{\text{PPM}}(d, M)$, by solving $\partial R_{\text{PPM}}(d, M)/\partial M = 0$, which yields

$$\frac{\ln(M^*)}{M^*+d} = \frac{1}{M^*}$$

or

$$M^* = \frac{d}{W(d/e)}. \tag{4}$$

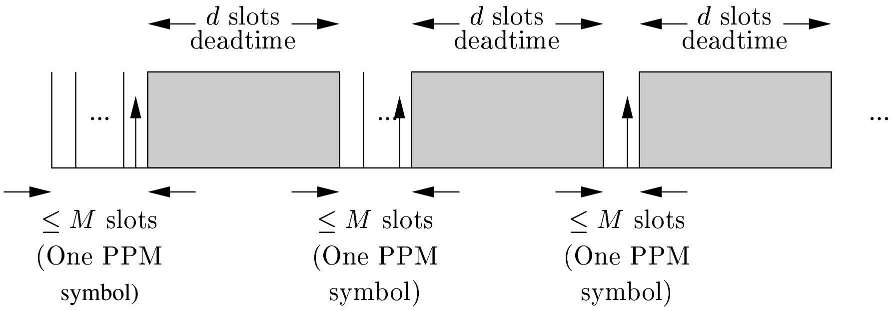

Figure 6: In TPPM signaling, the designated dead-time begins immediately after the pulse of the PPM symbol.

We will allow non-integer M in analysis to simplify expressions, since rounding has a negligible affect on rate for large d. However, all numerical results will be for integer M. Noting that $R_{\text{PPM}}(d, M^*) = \frac{1}{T_s M^*}$, the maximum rate is

$$R_{\text{PPM}}(d) = \frac{W(d/e)}{T_s d \ln 2} \text{ bits/s}$$

By an application of L'Hôpital's rule, one can show $E_{\text{PPM}}(d) \to 1$ as $d \to \infty$, i.e., PPM achieves capacity in the limit of large d. However, for d in our range of interest, significant gains in throughput over PPM are available, as illustrated in Table 1.

3.2 Truncated PPM with added dead-time

With PPM, there are 'unused' non-pulsed slot positions in the transmitted signal following each pulse– unused in the sense that they neither convey information, nor are necessary for satisfying the dead-time constraint. It would be more efficient for the $\log_2 M$ bits to map to a pulsed slot position and follow each pulse by exactly d non-pulsed slots. This signaling scheme is referred to as Truncated PPM (TPPM) [13], and is illustrated in Fig. 6. A graph and tree presenting the allowable TPPM code sequences is illustrated in Fig. 7.

Since the duration of a codeword mapping to $\log_2 M$ bits is variable, the code has an *average* rate

$$R_{\text{TPPM}}(d, M) = \frac{\ln(M)}{T_s(\frac{M+1}{2} + d) \ln 2} \text{ bits/s.}$$

For a given value of d we find M^*, the argument that maximizes $R_{\text{TPPM}}(d, M)$, by solving $\partial R_{\text{TPPM}}(d, M)/\partial M = 0$, which yields

$$M^* = \frac{2d + 1}{W(\frac{2d+1}{e})}.$$

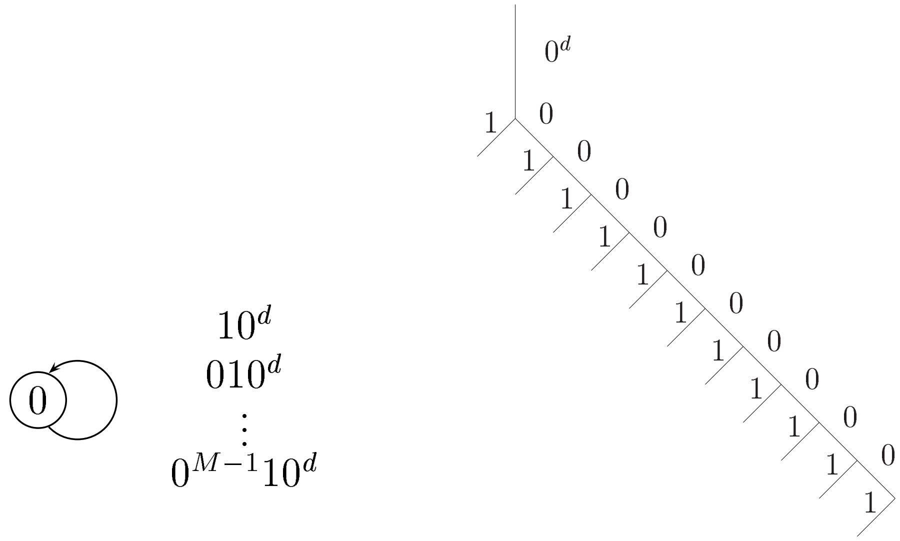

Figure 7: Graph and tree presenting TPPM code sequences.

Noting that $R_{\text{TPPM}}(d, M^*) = \frac{2}{T_s M^*}$, the maximum achievable rate is

$$R_{\text{TPPM}}(d) = \frac{2W(\frac{2d+1}{e})}{T_s(2d+1)\ln 2} \text{ bits/s}$$

Since $R_{\text{TPPM}}(d)$ is bounded above by $C(d)$ and below by $R_{\text{PPM}}(d)$, TPPM also achieves capacity in the limit of large d. TPPM is a low-complexity scheme that demonstrates significant throughput gains over PPM, e.g., Table 1. However, there are practical issues with implementing variable-rate decoders; these are addressed in Section 4.

3.3 Non-pulsed systems

PPM and TPPM with added dead-time both satisfy a (d, k) constraint for finite k. A small k is desirable for slot-timing recovery. Suppose, however, that a finite k is not required. In this section, we consider the efficiency of codes that allow arbitrarily long sequences without a pulse. To facilitate this, we describe a class of constrained systems, the (M, d, ∞) systems [14]. A sequence over an $M - ary$ alphabet $\{0, 1, \ldots, M-1\}$ belongs to an (M, d, ∞) constrained system if there are at least d but no more than k zeros between two nonzero symbols. The capacity of the system, in the sense of Eq. (1) is $C(M, d, \infty) = \log_2(\lambda)$ bits/symbol, where λ is the largest positive root of $(M-1)\lambda^{-(d+1)} + \lambda^{-1} - 1 = 0$.

The non-pulsed systems are introduced in this way as it will lead to simple descriptions of low-complexity, highly efficient codes.

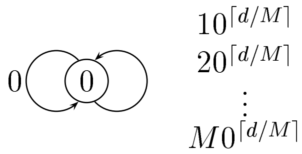

Figure 8: Graph presenting the $(M+1, \lceil d/M \rceil, \infty)$ system.

To satisfy an underlying (d, ∞) constraint, choose the system $(M+1, \lceil d/M \rceil, \infty)$ and map each symbol in the alphabet $\{0, 1, \ldots, M\}$ to an M-bit, or M-slot, word with a single one in the ith slot. With this mapping, the $(M+1, \lceil d/M \rceil, \infty)$ system, illustrated in Fig. 8, maps to a subset of the (d, ∞) system. The capacity of system $(M+1, \lceil d/M \rceil, \infty)$, which we will denote $C(M,d)$ is $C(M,d) = \frac{\ln(\lambda)}{M}$ nats/slot, where λ is the largest positive root of

$$M\lambda^{-(\lceil d/M \rceil + 1)} + \lambda^{-1} - 1 = 0. \tag{5}$$

For small $\lceil d/M \rceil$ exact solutions may be found efficiently for (5). For large $\lceil d/M \rceil$ substitute $\lambda = e^{MC(M,d)}$ and use the approximation $e^{-MC(M,d)} \approx 1 - MC(M,d)$ giving

$$C(M,d) \approx \frac{W\left(M(\lceil d/M \rceil + 1)\right)}{T_s M(\lceil d/M \rceil + 1)\ln 2} \text{ bits/s.} \tag{6}$$

Note that for $M = 1$ this reduces to (3).

Table 2 illustrates the trade-off of capacity versus M, the size of the alphabet, for the case $d = 512$. We also tabulate $C(M,d)/C(d)$, a measure of achievable efficiency. There exist capacity-achieving, low-complexity codes into $(M+1, \lceil d/M \rceil, \infty)$ systems for certain pairs (M, d). These codes are discussed in section 4.1.

3.4 Rate comparisons

Fig. 9 illustrates M^*, the optimum choice of M, for PPM and TPPM as a function of d. We observe M^* increases roughly linearly with d over the range plotted. Fig. 10 illustrates the efficiencies of fixed-orders of PPM , the optimal order TPPM, and STPPM as a function of d. The PPM order begins with 2 and increases in powers of two to 256. As noted before, the efficiencies of the schemes–when allowed to choose optimal order–will approach 1 for large d.

Suppose that T_d is fixed but T_s is a design parameter. Without a d constraint, a linear decrease in T_s would yield a linear increase in throughput. However, this is not the case in the presence of a d constraint. Fig. 11 illustrates the rates of PPM and TPPM and the capacity of the constraint as a function

M	$C(M,512)$	$C(M,512)/C(512)$
1	0.0132008	1
2	0.0131796	.998
4	0.0131795	.998
8	0.013137	.995
16	0.0130521	.989
32	0.012883	.976
64	0.0125487	.951
128	0.0119012	.902
256	0.0107203	.812
512	0.00885132	.671

Table 2: Capacity and achievable efficiency of $(M+1, \lceil d/M \rceil, \infty)$ systems that satisfy a $(512, \infty)$ constraint.

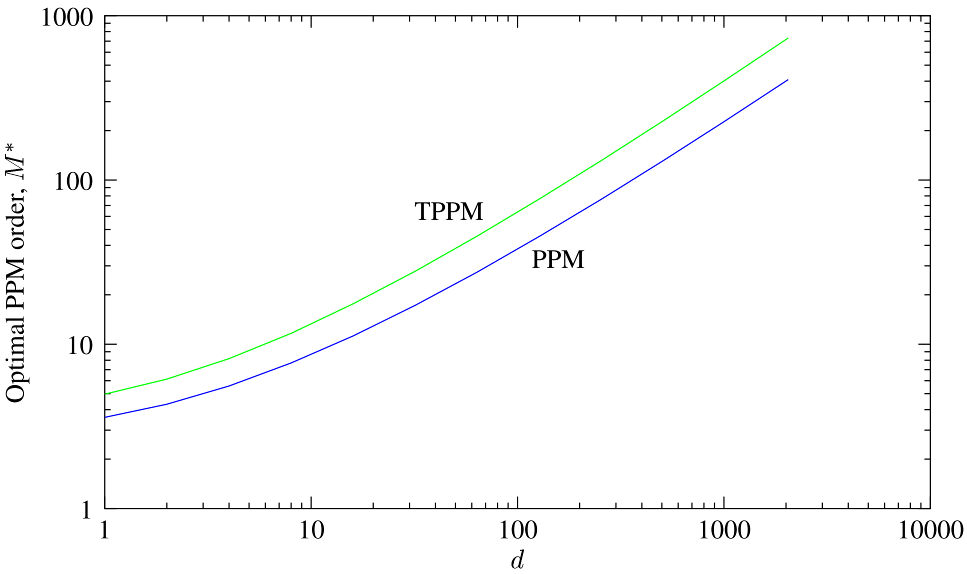

Figure 9: M^* for PPM and TPPM for (d, ∞) constraint.

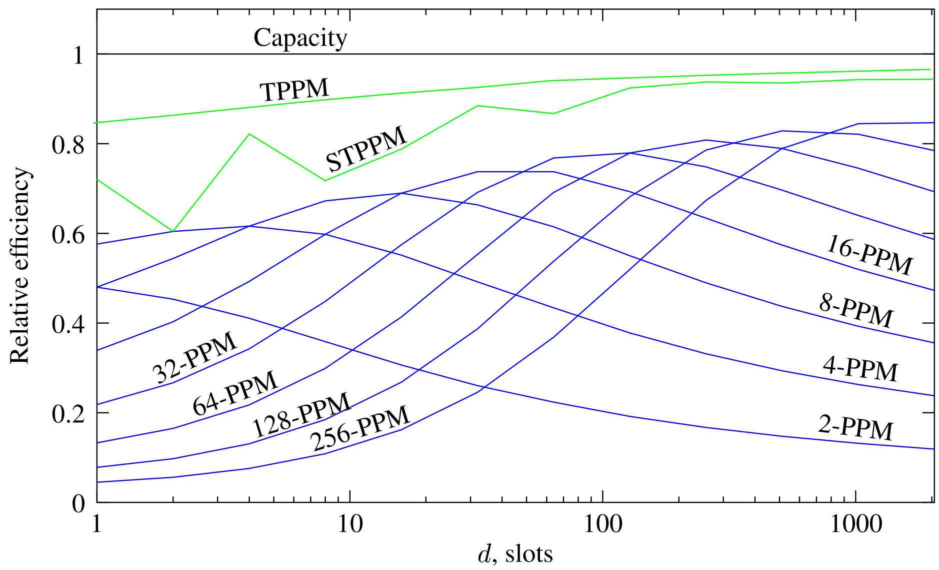

Figure 10: Efficiency of PPM for various orders, TPPM for optimal order, and STPPM

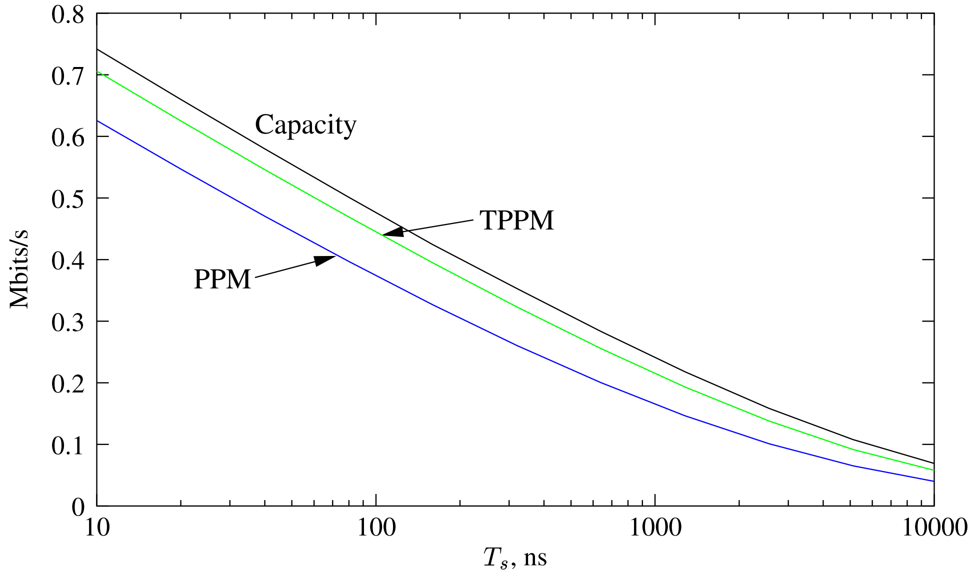

Figure 11: PPM, TPPM Rates and Capacity in Mbits/s

of T_s for $T_d = 10.24\mu s$. The throughput increases linearly for an exponential decrease in T_s, and the slopes in Figure 11 quickly approach–as T_s decreases–a constant $-\ln(10)/(T_d \ln(2)) \approx -.324$ Mbit/s, such that a 10-fold decrease in T_s is required to affect an increase of .324 Mbit/s in throughput.

4 Constrained Encoders and Decoders

4.1 Non-pulsed Position Modulation: efficient and low complexity

As noted in Section 3.3, with the appropriate $(M+1)$-ary alphabet, a code satisfying a $(M+1, \lceil d/M \rceil, \infty)$ constraint satisfies a (d, ∞) constraint. The following results were shown in [14, 15]:

Result 1 $\forall d \in [0, \infty)$ $C(M, d, \infty)$ *is rational only for* $\{M : M = 2^{dm}(2^m - 1) + 1, m$ *integer* $\}$

Result 2 $\forall d \in [0, \infty)$ *and* $\{M : M = 2^{dm}(2^m - 1) + 1, m$ *integer* $\}$, $C(M, d, \infty) = m$ *bits/symbol.*

Result 3 $\forall (M, d, \infty)$ *with rational capacity* C*, there exists a fixed-rate code into* (M, d, ∞) *with* $R = C$ *and* 2^{dm} *encoder states.*

Result 3 is significant in two respects: first, it shows the existence of capacity-achieving codes; second, the encoders have the fewest states for a fixed-rate code into the constraint with $R = C$. We will refer to capacity-achieving codes into $(M + 1, \lceil d/M \rceil, \infty)$ systems as Non-pulsed Position Modulation (NPM). From Result 1, the capacity is rational only for those pairs $(\lceil d/M \rceil, M)$ such that

$$M = 2^{\lceil d/M \rceil m}(2^m - 1)$$

for integer m. Taking $m = 1$ and assuming integer d/M, we have

$$M = \frac{d \ln 2}{W(d \ln 2)} \tag{7}$$

Table 3 lists all pairs (d, M) with integer d/M, $d \leq 2^{12}$, that satisfy (7). There is an NPM code corresponding to each entry with $2^{d/M}$ encoder states. The efficiency of the code relative to the (d, ∞) constraint is $E_{\text{NPM}}(d) = C(M, d)/C(d)$.

Note that for integer d/M, we have the exact solution

$$C(M, d) = \frac{W(d \ln 2)}{T_s d \ln 2} \text{bits/s,}$$

d	M	number of encoder states	E_{NPM}
2	2	2	.9067
8	4	4	.8968
24	8	8	.9050
64	16	16	.9158
160	32	32	.9281
384	64	64	.9352
2048	256	256	.9472

Table 3: Efficiency and complexity for NPM

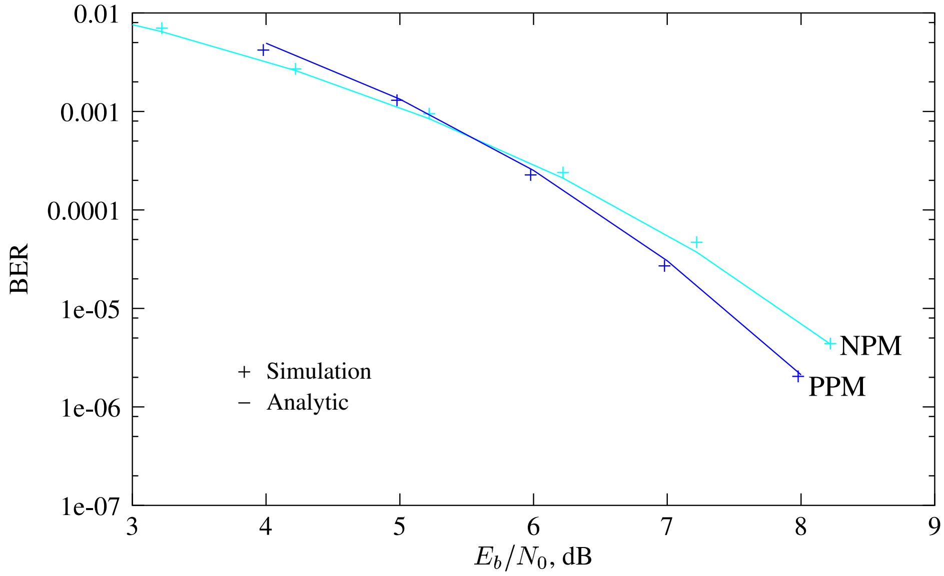

Figure 12: NPM and PPM, $d = 64$

which we can compare to the approximation (6).

Fig. 12 illustrates the performance of an NPM code relative to PPM for $d = 64$. The NPM code has a lower average transmitted energy: 1 pulse per ≈ 96 slots compared to 1 pulse per 80 slots for PPM, and a higher throughput: 1 bit per 16 slots compared to 1 bit per 20 for PPM. Hence NPM delivers 6 bits per pulse compared to 4 bits per pulse for PPM. However, since the NPM code allows an all-zeros sequence, the minimum Euclidean distance is 1, half the minimum distance of PPM. This yields a net loss in performance as a function of E_b/N_0–setting aside throughput gain–for a specified d in spite of the throughput and average energy gains. This loss, however, may be returned in a concatenated coding scheme. The larger problem with NPM codes is the difficulty of acquiring synchronization.

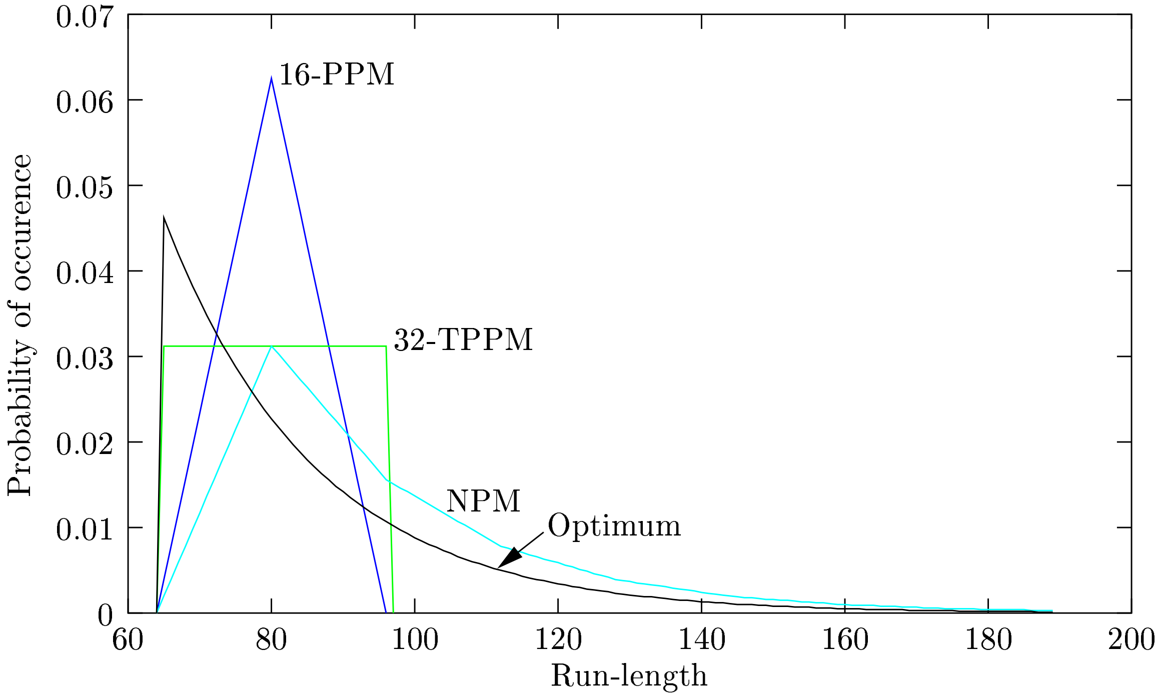

Figure 13: Distribution of run-lengths in PPM, TPPM, NPM, and optimal for $d = 64$

Any sequence in a (d, ∞) system may be uniquely parsed into patterns, or phrases, 0^j1 where $j \geq d$. The phrases 0^j1 are referred to as *run-lengths* and their distribution will have an impact on slot-synchronization schemes. Fig. 13 illustrates the distribution of lengths of run-lengths for the case $d = 64$ of the NPM code described above, 16-ary PPM, 32-ary TPPM, and the optimal, capacity achieving, distribution.

4.2 TPPM: variable rate implementation issues

TPPM is an attractive low complexity, high throughput scheme. However, it has two implementation issues that are common to variable-rate schemes. The first is the difficulty of adapting decoding algorithms to function with variable-rate codes. It may be particularly difficult to accommodate the code as an inner code in a concatenated coding scheme since a block of data from the outer code would map to a variable length block of transmitted symbols.

The second problem is the possibility of catastrophic error propagation in decoding due to loss of frame synchronization. Assume the TPPM decoder operates similar to an PPM decoder, by choosing the maximum slot count in an appropriate window. If an error is made in the estimation of the pulse position, the location of the following window will be incorrect. The detector will, however, re-synchronize with the next correctly detected pulse position. In Appendix A we show the probability of re-synchronizing in the frame following a pulse position estimation is $\approx 2/3$ for q small and M large.

There may be methods of averting the problems with TPPM by buffering data and performing an appropriate sequence-detection algorithm. However, given lower-complexity options with similar performance, we did not pursue implementing TPPM.

4.3 Permutation Modulation Codes

Permutation codes were introduced by Slepian [16], and were later proposed for constructing high-rate (d, k) codes by Wolf [17]. An application of a permutation code for an optical channel was described in [18], although not subject to a d constraint. Permutation codes are attractive as they have simple Maximum Likelihood decoders, albeit with complex encoders. Wolf showed that permutation codes asymptotically achieve the capacity of a (d, k) constraint, but didn't address the complexity of encoding. Recent work by Datta and McLaughlin [19] has addressed issues of encoding complexity. Permutation codes are an attractive option for implementing our constraints for the reasons mentioned; namely they can be made arbitrarily efficient, and have a low-complexity ML-decoding. We will see, however, that the complexity for our range of d prohibits their use. We will give a brief description in order to quantify the encoder complexity.

A sequence in the (d, ∞)-system may be uniquely parsed into phrases 0^j1 where $j \geq d$. To construct a permutation code, choose a *phrase profile vector* $\mathbf{v} = (v_1, v_2, \ldots, v_N)$ where each v_i is an allowed phrase and phrases may be repeated. The permutation codebook U corresponding to $\mathbf{v}$ consists of all distinct permutations of $\mathbf{v}$. Clearly U may be used to construct a code into the (d, ∞) constraint.

Let n_j be the number of times phrase $0^{j-1}1$ appears in $\mathbf{v}$. Then all codewords have length $L = \sum_{j=d+1}^{\infty} j n_j$ and

$$|U| = \frac{N!}{\prod_{j=d+1}^{\infty} n_j!}.$$

We can design a code with rate $R_{PC} = \log|U|/(LT_s)$ arbitrarily close to $C(d)$ by taking N sufficiently large and choosing a phrase profile vector with $n_j/N \approx 2^{-jC(d)}$. However, for $C(d)$ small, N must be chosen very large for the code to be efficient. Table 4 lists the parameters of the permutation code with the smallest N such that the efficiency exceeds that of PPM (using the optimal choice of M from (4)). We see that the size of the codebook for a competitive complexity rules out permutation codes. This conclusion doesn't take into account the error-correcting capability of a permutation code, which would be significantly better than PPM. However, this capability would be more efficiently obtained by

d	codebook size $\|U\|$	codeword length $\log_2 L$	phrase profile length N	$E_{PC}(d)$
64	$2^{278} \approx 5 \times 10^{83}$	12.4	65	.773
128	$2^{548} \approx 9 \times 10^{164}$	14.1	109	.792
256	$2^{1091} \approx 3 \times 10^{328}$	15.8	188	.811
512	$2^{2171} \approx 3 \times 10^{653}$	17.6	329	.829
1024	$2^{4354} \approx 5 \times 10^{1310}$	19.4	586	.844

Table 4: Permutation Codes

concatenating a lower complexity outer error-correcting code.

4.4 Synchronous Variable-Length Codes

The encoders considered so far have been either fixed rate or variable rate. Allowing a variable rate adds a degree of freedom in design, resulting in higher efficiency and/or lower complexity encoders. However, variable rate encoding and decoding has practical drawbacks. A compromise is to allow a synchronous rate, namely mappings of mp bits to mq bits, where p, q are fixed positive integers, and m is a positive integer that can vary. Methods of constructing synchronous variable-length codes were initially described in [20], and reviews of various approaches may be found in [9, 21].

We describe a new systematic procedure to construct synchronous encoders and decoders for (d, ∞) constraints. The procedure may be interpreted as a practical method of approaching rates of TPPM, suggested by Dolinar's 'partially-padded-PPM' [22]. Choose a rate $p/q < C(d)$ bits/slot. We desire a set of variable-length codewords $\mathcal{C} = \{c_1, c_2, \ldots, c_N\}$ such that any sequence formed by freely concatenating the codewords satisfies the constraint, the codeword lengths $l(c_i)$ are multiples of q, no codeword is the prefix of another (sufficient but not necessary to guarantee decodability), and the collection satisfies the Kraft (In)equality

$$\sum_{c_i \in \mathcal{C}} 2^{-l(c_i)p/q} = 1. \tag{8}$$

We can use such a set to construct a synchronous variable-length code mapping unconstrained binary sequences into the constraint.

We detail one method to construct such a set that leads to a low-complexity encoder and decoder. The codewords are constructed as nodes on a binary tree. The root of the tree is the pattern 0^d. Branches with a label 1 are extended with zeros to the first length that is a multiple of q. At this point, the branch

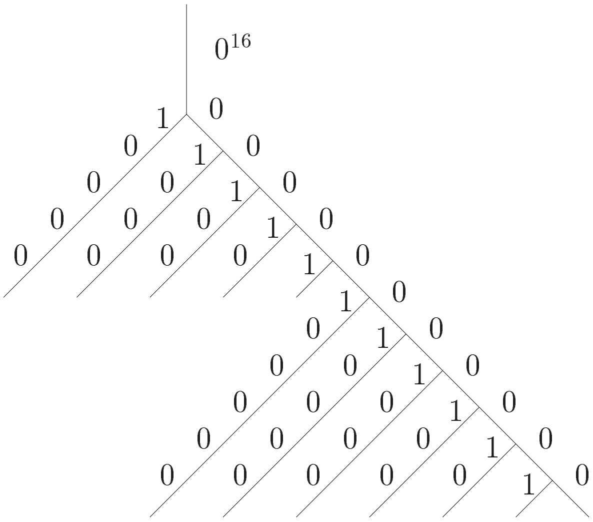

Figure 14: STPPM construction for $(d, k) = (16, \infty)$, $q = 7$

label is taken as a codeword. The tree is expanded until we have a set of codewords that satisfies (8). Fig. 14 illustrates the procedure for $(d, k) = (16, \infty)$, $q = 7$.

The all zeros pattern is not allowed as a codeword, since allowing it reduces the minimum Euclidean distance from 2 to 1, the small gain in throughput does not offset the loss in distance (allowing the all zeros codeword does yield significant throughput gains for small d), and a finite k constraint is desired for synchronization. The encoding and decoding may be done at a fixed rate by using encoders and decoders with appropriate memory. Codes constructed via this method will be referred to as Synchronous Truncated Pulse Position Modulation (STPPM). A simple encoder implementation exists if we allow variable-out-degree states. A rate $1/7$ variable-out-degree encoder/decoder trellis corresponding to Fig. 14 is described in Appendix C.

This procedure does not allow rates arbitrarily close to capacity. One can show a rate p/q encoder may be constructed via this method into a (d, ∞) constraint if $K(q, d, p) \geq 1$, where

$$K(q, d, p) = 2^{-lp}\left(lq - d + \frac{q-1}{2^p - 1}\right),$$

and $l = \lfloor d/q \rfloor + 1$. A simple encoder/decoder trellis may be constructed if variable-out-degree states are allowed. An encoder using variable-out-degrees states exists with

$$m = \left\lceil \frac{1}{p} \log_2 \left(\frac{q-1}{(K(q, d, p) - 1)(2^p - 1)} \right) \right\rceil$$

d	q	states	edges	E_{STPPM}
16	7	4	14	.788
32	10	8	51	.885
64	17	6	41	.867
128	27	10	146	.924
256	46	12	301	.938
512	81	11	385	.935

Table 5: STPPM Variable-out-degree Encoder Parameters

states and no more than

$$mq + l - 1 - d$$

edges. Table 5 lists the parameters of a number of codes for a range of d of interest where in each case $p = 1$. The encoder/decoder complexity may be traded off for efficiency in a systematic manner by specifying a lower rate. Note that there are fewer than $q + 1$ distinct edge labels in each trellis stage.

The STPPM codes demonstrate throughput gains of 11%–17% over PPM. However, this may come at the cost of a lowering the bits transmitted per pulse. Fig. 15 illustrates the performance of an STPPM code relative to 8-PPM and 16-PPM for $d = 16$. The minimum distance of the three codes is the same, and the throughput of the PPM schemes are the same, whereas STPPM has a 14% higher throughput. The performance is differentiated due to the energy per bit requirements. STPPM transmits an average 3.375 bits per pulse whereas 16-PPM transmits 4 bits per pulse–yielding a net gain relative to STPPM of $\approx .74$dB–and 8-PPM transmits 3 bits per pulse–for a net loss relative to STPPM of $\approx .5$dB.

The efficiency is measured relative to a (d, ∞) constraint. However, the codes all impose a maximum run-length constraint–necessary for timing recovery and desired for distance properties (a fairer measure of efficiency would be relative to the appropriate (d, k) constraint). The distribution of run-lengths will impact the performance of timing-recovery schemes. Fig. 16 illustrates the distribution of lengths of run-lengths for PPM, TPPM, one implementation of a STPPM code and the optimum distribution at $d = 16$.

4.5 Finite-State Constrained codes

The literature on finite-state constrained codes treats much smaller values of d. Many of the design issues have received attention and there are several good tutorials on code construction, e.g., [8, 9, 21, 23]. Here we tabulate upper and lower bound on the complexity of finite-state code trellises, measured via the number of states, using bounds established in [24]. Table 6 illustrates the results, comparing them to parameters for variable and fixed out-degree STPPM trellises. The rate in each case is $1/q$.

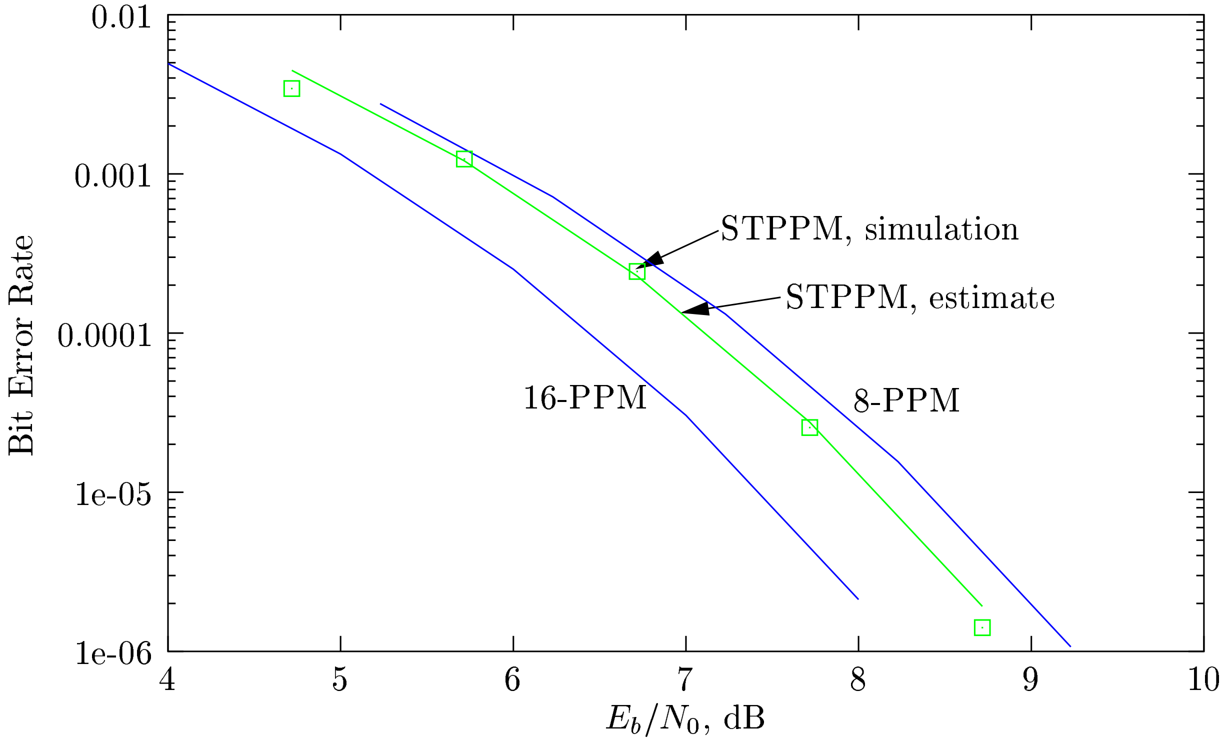

Figure 15: STPPM, 16-PPM, and 8-PPM, $d = 16$

The complexity of the finite-state code is for a code into the (d, ∞) constraint, not into a (d, k), $k < \infty$ constraint. Several rates are tabulated for $d = 128$ and 256 to illustrate trade-offs of complexity for efficiency. The upper bound is from an existence proof and is generally loose. It is not known how tight the lower bound is.

4.6 Complexity comparisons

As seen in Fig. 17, the STPPM codes demonstrate throughput (measured in bits per second) gains of 11%–17% over PPM. However, this may come at the cost of higher complexity, as seen in Fig. 18, where FS is the lower bound on the complexity of a finite-state construction.

5 The Constrained Code in a Concatenated Coding Scheme

The larger coding structure will use the constrained code in concert with an Error Correcting Code (ECC). The codes will be concatenated serially, as illustrated in Fig. 1. A bit interleaver may be inserted between the codes, serving to disperse error bursts in decoding, and providing particular performance

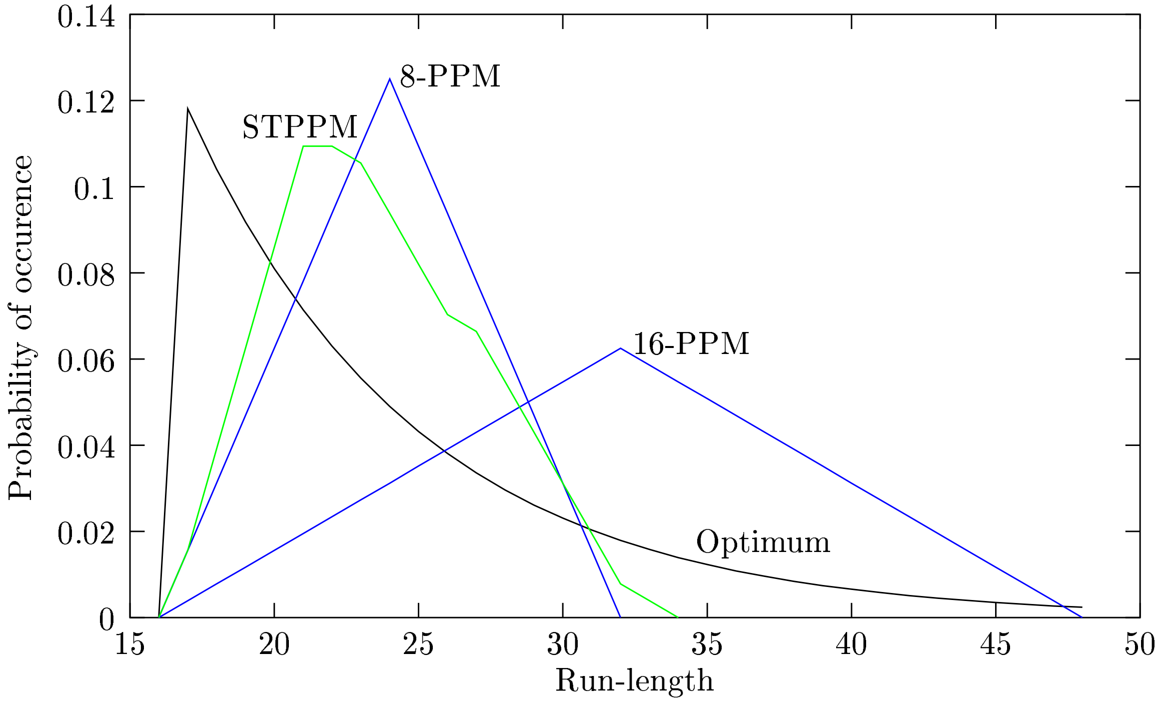

Figure 16: Runlengths distributions, $d = 16$

improvement in iterative decoding schemes. We will use the notation $\mathcal{C}_o \rightarrow \mathcal{C}_i$ to denote the non-iteratively decoded serial concatenation of outer code $\mathcal{C}_o$ and inner code $\mathcal{C}_i$. Iteratively decoded codes are denoted by $\mathcal{C}_o \leftrightarrow \mathcal{C}_i$. Iterative decoding follows the description in [25].

The concatenated order in Fig. 1 was not a foregone conclusion. Alternative concatenations of the codes may be considered and various concatenations of a constrained code with an ECC have been addressed for magnetic and optical storage channels, e.g., [26, 27]. The variations are proposed largely due to the high rates of constrained codes for storage–typically $m/(m+1)$, where m is on the order of $10 - 20$. Such high rate codes have poor error-correction capabilities, high decoding complexity and considerable error-propagation. Hence there is some motivation for placing them outside of an ECC. However, for our applications the very low rates of the constrained code favor the concatenation in Fig. 1.

The baseline system is taken to be RS$(M-1, k) \rightarrow M$-PPM, where RS$(M-1, k)$ denotes a rate $k/(M-1)$ Reed-Solomon code and the PPM demodulator produces hard-decisions. (Here, k does not refer to the run-length constraint.) Prior work investigated the system PCCC $\rightarrow M$-PPM [28], where PCCC is an iteratively decoded Parallel Concatenated Convolutional Code and the PPM demodulator produces soft-decisions—although it is not included in iterations. Peleg and Shamai [29] investigated the system PCCC $\leftrightarrow M$-PPM on a discrete-time Rayleigh-fading model, illustrating performance 1–2

d	q	Number of states: Finite-state Lower Bound	Finite-state Upper Bound	Variable out-degree	Fixed out-degree
16	7	4	16	4	10
32	10	8	64	8	36
64	17	12	160	6	35
128	28	20	448	7	51
	27	25	688	10	130
256	48	32	2448	8	112
	46	44	2940	12	270
512	81	73	8887	11	342

Table 6: Finite-state-code Parameters

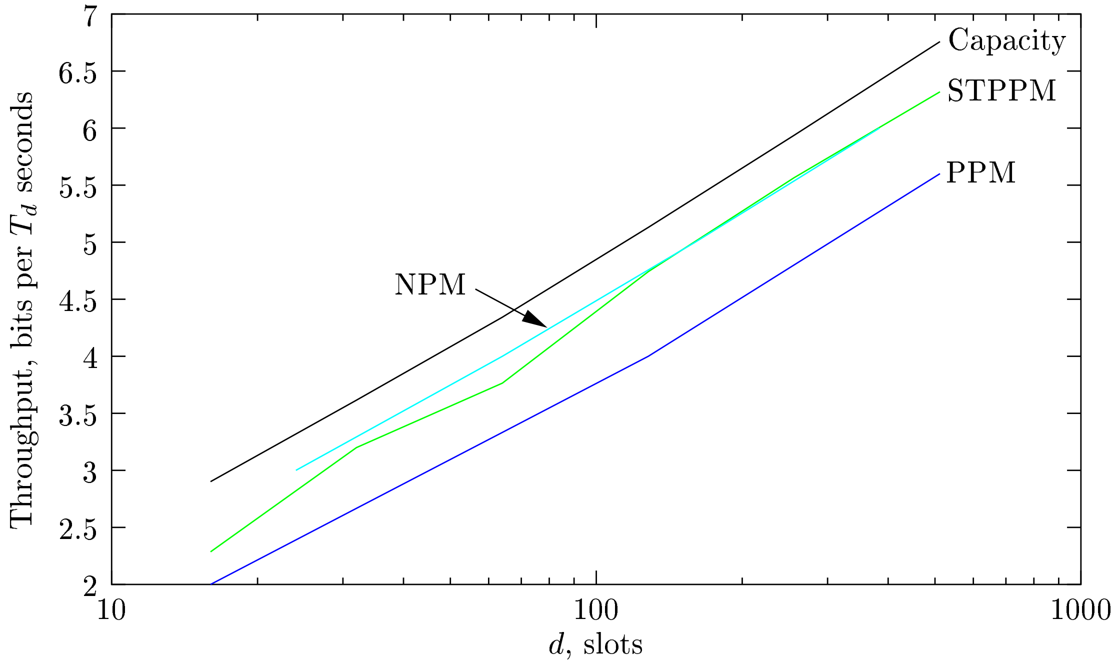

Figure 17: Throughput

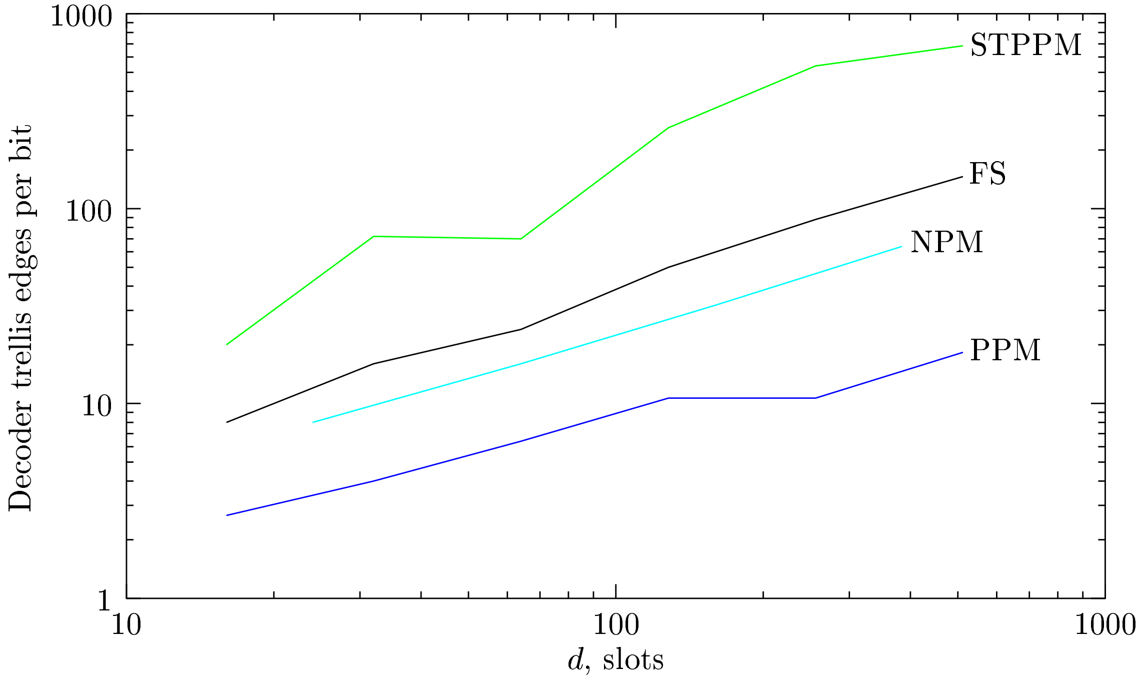

Figure 18: Complexity

dB from capacity.

We did not simulate the system RS$(n, k) \rightarrow$ STPPM, which would replace hard-decision PPM with a higher throughput constrained code. Since hard decisions are sufficient, the low-complexity ML decoder described by a VOD trellis could be used to decode STPPM. In this case, there is a trade-off of average transmitted energy and complexity for throughput.

5.1 Simulation Results

Fig. 19 illustrates performance under a $d = 16$ constraint. Uncoded 16-PPM satisfying $(d, k) = (16, 46)$, a rate $1/7$ STPPM code satisfying $(d, k) = (16, 32)$, RS$(15, 9) \rightarrow$ 16-PPM, CC$(3, 1/2) \leftrightarrow$ 16-PPM and CC$(3, 1/2) \leftrightarrow$ STPPM are illustrated, where CC$(3, 1/2)$ is the 4-state convolutional code with generator polynomial $g(D) = [1(1 + D^2)/(1 + D + D^2)]$. The capacity limits for throughput $(1/14)$ bits/slot on a hard and soft-decision channel constrained to use a 16-ary orthogonal signal set, e.g., 16-PPM are also illustrated. These curves represent the theoretical limits of a channel using 16-PPM with hard and soft decisions, respectively.

Both CC$(3, 1/2) \leftrightarrow$ 16-PPM and CC$(3, 1/2) \leftrightarrow$ STPPM used a 512-bit interleaver and 8 iterations. They illustrate gains of $\approx$ 2.5dB over RS$(15, 9) \rightarrow$ 16-PPM at a bit-error-rate of 10^{-5}. Simulation results demonstrate that 4 iterations would be sufficient at the higher values of SNR. A small additional

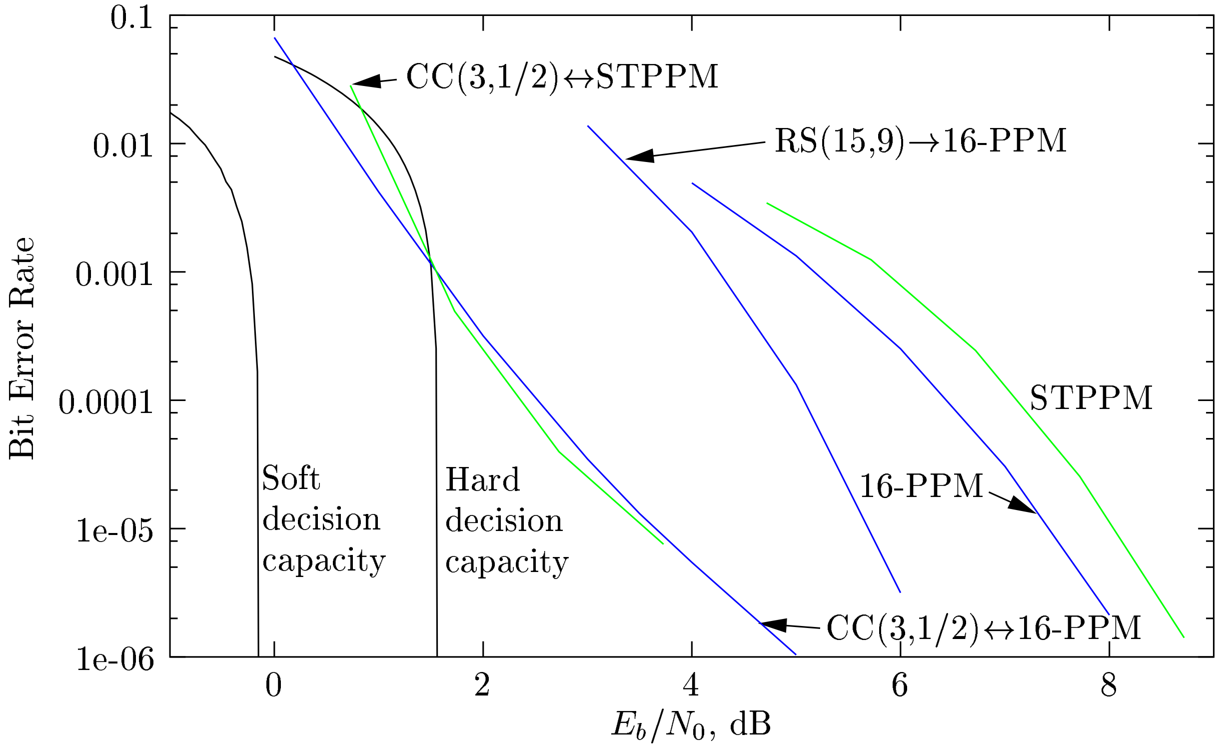

Figure 19: BER performance comparisons, $d = 16$.

gain of approximately 0.2dB was found for a 4096-bit interleaver with CC$(3, 1/2)$ $\leftrightarrow$ 16-PPM. Note that CC$(3, 1/2)$ $\leftrightarrow$ STPPM has a higher throughput ($1/14$ bits/slot) than CC$(3, 1/2)$ $\leftrightarrow$ 16-PPM ($1/16$ bits/slot).

Fig. 20 illustrates performance under a $d = 1024$ constraint. Uncoded 256-PPM satisfying $(d, k) = (1024, 1534)$, RS$(255, 128)$ $\rightarrow$ 256-PPM, and CC$(3, 1/2)$ $\leftrightarrow$ 256-PPM for two interleaver sizes are illustrated. The capacity limits for throughput $1/320$ bits/slot on a hard and soft-decision channel constrained to use a 256-ary orthogonal signal set are also illustrated. CC$(3, 1/2)$ $\leftrightarrow$ 256-PPM with a 4096-bit interleaver and 8 iterations shows gains of 2.3dB over RS$(255, 128)$ $\rightarrow$ 256-PPM at a bit-error-rate of 10^{-5}. CC$(3, 1/2)$ $\leftrightarrow$ 256-PPM performs 0.4dB better than *any* system with the same throughput that uses hard-decision 256-PPM.

These results are surprising in light of the low-complexity of the constituent codes and lack of recursiveness in the inner code. They provide a strong argument for replacing the baseline RS$(M - 1, k)$ $\rightarrow$ M-PPM with a low-complexity ECC serially-concatenated with PPM or some other constrained code.

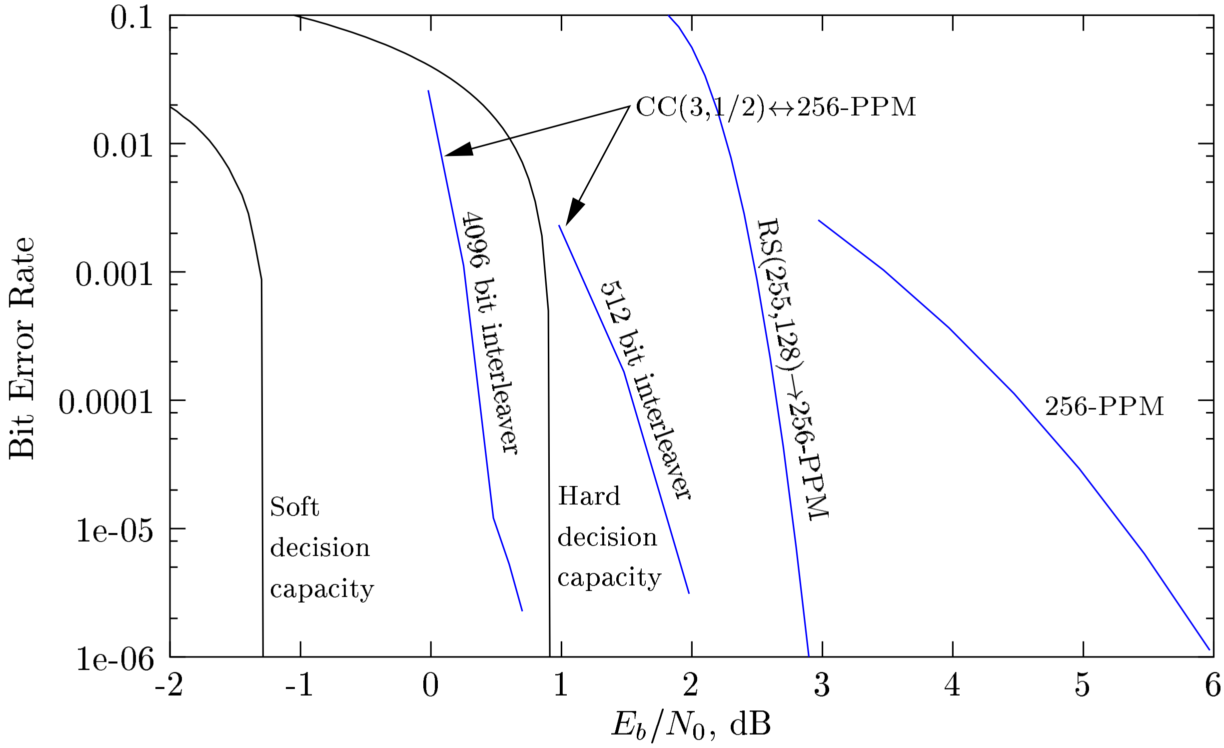

Figure 20: BER performance comparisons, $d = 1024$.

6 Conclusions/Future Work

There are certain trade-offs in replacing PPM with STPPM or another constrained code. The constrained codes considered provide higher throughput at the cost of increased complexity. Whether the code gains in energy per transmitted bit and run length distribution depends on implemented parameters.

The gains of the concatenated, iteratively decoded schemes over the baseline RS→PPM are more clear. We have illustrated that low-complexity iterative schemes provide significant gains over the baseline. We expect to improve on these gains with a better understanding of the interaction between the outer code and constrained code. For example, recent results that include an accumulator—a $1/(1+D)$ mapping—prior to the PPM mapping in order to add recursiveness to the mapping have shown significant additional gains for small d.

We conjecture that the interleaver in the serial concatenated system need only be large enough to distribute each bit in the most likely error bursts from the outer code into distinct PPM symbols. We expect that interleavers larger than this will show only small improvements.

The iteratively decoded schemes do, however, rely on statistics from each signal slot. This may be infeasible at the proposed operating rates. Future work will investigate the degradation in performance when approximations to complete slot statistics are used.

A Frame re-synchronization in TPPM

Let $a_i \in \{1, \ldots, M\}$ denote the i-th transmitted M-ary TPPM symbol, $\hat{a}_i$ the estimate of a_i at the detector, $s_i = s_{i-1} + a_i + d$ the location of the i-th frame, and $\hat{s}_i = \hat{s}_{i-1} + \hat{a}_i + d$ the corresponding estimate of s_i. Suppose the detector is initially synchronized, $\hat{s}_i = s_i$, and an error is made in the estimate of the current slot, $\hat{a}_i \neq a_i$. Let q denote the probability of this event, and assume that incorrect slots are chosen equally likely.

If $\hat{a}_{i+1} = a_{i+1} - \hat{a}_i + a_i$, i.e., the next signal slot occurs in the following window and is detected correctly, then $\hat{s}_{i+2} = s_{i+2}$, and the detector is synchronized. The detector will re-synchronize in k frames where k is the first integer such that

$$\hat{a}_{i+k} - a_{i+k} = -\sum_{j=1}^{k-1} \hat{a}_{i+j} - a_{i+j}.$$

Let $e \stackrel{\text{def}}{=} a(i) - \hat{a}_i$, hence $q = P[e \neq 0]$. Assuming non-signal slots are chosen equally likely, e has distribution

$$p(e) = \begin{cases} \frac{q}{M(M-1)}(M - |e|) & , e \neq 0 \\ 1 - q & , e = 0 \end{cases}.$$

The probability of a frame re-synchronization in the frame following a loss of sync is

$$P[\hat{a}_{i+1} - a_{i+1} = e | e \neq 0] = P[\hat{a}_{i+1} - a_{i+1} = e, e \neq 0]/q.$$

The problem reduces to finding the joint probability

$$\begin{aligned} P[\hat{a}_{i+1} - a_{i+1} = e, e \neq 0] &= \sum_{(\hat{a}_{i+1}, a_{i+1}, e): e \neq 0, \hat{a}_{i+1} - a_{i+1} = e} p(\hat{a}_{i+1}, a_{i+1}, e) \\ &= \sum_{(\hat{a}_{i+1}, a_{i+1}, e): e \neq 0, \hat{a}_{i+1} - a_{i+1} = e} p(a_{i+1}) p(e | a_{i+1}) p(\hat{a}_{i+1} | a_{i+1}, e) \\ &= \frac{q}{M^2(M-1)} \sum_{(\hat{a}_{i+1}, a_{i+1}, e): e \neq 0, \hat{a}_{i+1} - a_{i+1} = e} (M - |e|)(1 - q) \\ &= \frac{q(1-q)}{3M}(2M - 1). \end{aligned}$$

B BCJR on edges

A block of information symbols $\mathbf{u} = (u_1, \ldots, u_N)$ is encoded by a code $\mathcal{C}$ to yield a codeword $\mathbf{c} = \mathcal{C}(\mathbf{u}) = (c_1, \ldots, c_N)$, where u_k, c_k are binary-vectors. The code is described by a time-invariant trellis (straight-forward modifications would treat time-varying trellises) 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, $i(e)$, and terminal state, $t(e)$. Let s_k denote the state at time k, so that $t(e_{k-1}) = s_k = i(e_k)$, let $u(e)$ and $c(e)$ denote the input and output edge labels such that $u_k = u(e_k), c_k = c(e_k)$. Throughout we use the notation $u_{i:j} \stackrel{\text{def}}{=} (u_i, u_{i+1}, \ldots, u_j)$, and shorthand $P\{u\} = P\{u_k = u\}$ when this is clear from context.

A symbol sequence $\mathbf{y} = (y_1, \ldots, y_N)$ is observed, where $\mathbf{y}$ is conditionally independent of prior observations and edges given the previous edge, i.e., $P\{y_{k:k+m}|e_j, y_j; j \leq k-1\} = P\{y_{k:k+m}|e_{k-1}\}$.

It is assumed we have estimates

$$\begin{aligned} P_k(u; I) &\stackrel{\text{def}}{=} \hat{P}\{u_k = u\}, \\ P_k(c; I) &\stackrel{\text{def}}{=} \hat{P}\{c_k = c\}, \end{aligned}$$

of the distributions on $\mathbf{u}$ and $\mathbf{c}$, where I denotes that this is a-priori, or input, information. Given the observation $\mathbf{y}$, and knowledge of $\mathcal{C}$, we desire to compute the quantities

$$\begin{aligned} P_k(u; O) &\stackrel{\text{def}}{=} P\{u_k = u|\mathbf{y}\} \qquad (9) \\ P_k(c; O) &\stackrel{\text{def}}{=} P\{c_k = c|\mathbf{y}\}, \end{aligned}$$

where O denotes that this is a-posteriori, or output, information. These are obtained by computing

$$\begin{aligned} P\{u_k = u, \mathbf{y}\} &= \sum_{e:u(e)=u} P\{e_k = e, \mathbf{y}\} \\ P\{c_k = c, \mathbf{y}\} &= \sum_{e:c(e)=c} P\{e_k = e, \mathbf{y}\}, \end{aligned}$$

hence we need to determine

$$\lambda_k(e) \stackrel{\text{def}}{=} P\{e_k = e, \mathbf{y}\}$$

for each edge in the trellis. To this end, define

$$\begin{aligned}
\alpha_k(e) &\stackrel{\text{def}}{=} P\{e_k = e, y_{1:k}\} \\
\beta_k(e) &\stackrel{\text{def}}{=} P\{y_{k+1:N} | e_k = e\} \\
\gamma_k(e) &\stackrel{\text{def}}{=} P\{e_k = e, y_k | i(e_k) = i(e)\} \\
&= P\{e_k = e | i(e_k) = i(e)\} P\{y_k | e_k = e\},
\end{aligned}$$

and note that

$$\begin{aligned}
\lambda_k(e) &= P\{e_k = e, y_{1:k}\} P\{y_{k+1:N} | e_k = e, y_{1:k}\} \\
&= \alpha_k(e)\beta_k(e)
\end{aligned}$$

$$\begin{aligned}
\alpha_k(e) &= \sum_{e' \in \mathcal{E}} P\{e_{k-1} = e', e_k = e, y_{1:k-1}, y_k\} \\
&= \sum_{e' \in \mathcal{E}} P\{e_{k-1} = e', y_{1:k-1}\} P\{e_k = e, y_k | e_{k-1} = e'\} \\
&= \sum_{e' : t(e') = i(e)} \alpha_{k-1}(e') \gamma_k(e) \\
&= \gamma_k(e) \sum_{e' : t(e') = i(e)} \alpha_{k-1}(e')
\end{aligned}$$

$$\begin{aligned}
\beta_k(e) &= \sum_{e' \in \mathcal{E}} P\{e_{k+1} = e', y_{k+2:N}, y_{k+1} | e_k = e\} \\
&= \sum_{e' \in \mathcal{E}} P\{y_{k+2:N} | e_{k+1} = e'\} P\{e_{k+1} = e', y_{k+1} | e_k = e\} \\
&= \sum_{e' : i(e') = t(e)} \beta_{k+1}(e') \gamma_{k+1}(e')
\end{aligned}$$

The algorithm is initialized by setting

$$\begin{aligned}
\alpha_1(e) &= P\{e_1 = e\} P\{y_1 | e_1 = e\} \\
\beta_{N-1}(e) &= \sum_{e' : i(e') = t(e)} \gamma_N(e'),
\end{aligned}$$

or, equivalently, $\beta_N(e) = 1$ (β_N is not required as $\lambda_N(e) = \alpha_N(e)$).

If all outgoing edges from a state have distinct input labels, then

$$\begin{aligned} P\{e_k = e | i(e_k) = i(e)\} &= P\{u_k = u(e)\} \\ &= P_k(u(e); I). \end{aligned}$$

We encounter two common cases of the observation $\mathbf{y}$. Suppose the observation is $\mathbf{y} = \mathbf{c} + \mathbf{n}$, where $\mathbf{n} \sim \mathcal{N}(0, \sigma^2 I)$, e.g., the output observed by transmitting the coded sequence over an AWGN channel–extensions to other channels are straightforward. Then

$$\begin{aligned} P_k(c(e); I) &= K \exp(-||y_k - c(e)||^2/(2\sigma^2)) \\ &= P\{y_k | e_k = e\} \end{aligned}$$

where K is a constant and $|| \cdot ||$ denotes the Euclidean norm. Alternately, suppose the observation is $\mathbf{y} = \mathbf{c}$, e.g., the estimated output from an inner code. Then

$$\begin{aligned} P\{y_k | e_k = e\} &= P\{c_k = c(e)\} \\ &= P_k(c(e); I). \end{aligned}$$

In either case, the observed and a priori information enters the algorithm via the term

$$\gamma_k(e) = P_k(u(e); I) P_k(c(e); I).$$

Finally, we have

$$\begin{aligned} P_k(u; O) &= \frac{\sum_{e:u(e)=u} \lambda_k(e)}{\sum_{e \in \mathcal{E}} \lambda_k(e)} \\ P_k(c; O) &= \frac{\sum_{e:c(e)=c} \lambda_k(e)}{\sum_{e \in \mathcal{E}} \lambda_k(e)}. \end{aligned}$$

Note that multiplying each $\lambda_k(e)$ by a constant independent of e will not affect the result. Hence one can multiply the α's, β's, or γ's by a constant for each k to simplify computation or prevent over or under flow. Bit probabilities, rather than symbol probabilities, follow via some straight-forward extensions, e.g. [25].

B.1 Computational Complexity

The following straightforward simplifications are made to the algorithm to yield a rough analysis of decoding complexity. Note that $\beta_k(e)$ depends only on $t(e)$, hence we can compute and store one value per state. Similarly, the sum in computing $\alpha_k(e)$ need only be determined once for each state. The term $\sum_{e\in\mathcal{E}} \lambda_k(e)$ can be computed once and $\gamma_k(e)$ need only be computed for the distinct pairs $(P_k(u(e);I), P_k(c(e);I)$–typically one of these distributions is assumed uniform, hence the cost of computing $\gamma_k(e)$ is proportional to the size of the associated input or output alphabet. By computing the β_k's prior to the α_k's, the storage requirements are only negligibly increased over a state-based algorithm.

Let $\mathcal{A}$ denote the input alphabet, i.e. $u_k \in \mathcal{A}$. The algorithm and an approximation of associated costs for computing $P_k(u;O)$ are listed in Table 7

	Step	Cost
1.	Receive $\mathbf{y}$	none
2.	Determine $P_k(u;I), P_k(c;I)$	varies
3.	Compute $\gamma_k(e)$	$N\lvert\mathcal{A}\rvert$
4.	Compute $\beta_k(t(e))$	$N\lvert\mathcal{E}\rvert$ multiplies
		$N(\lvert\mathcal{E}\rvert - \lvert\mathcal{V}\rvert)$ adds
5.	Compute $\sum_{e':t(e')=i(e)} \alpha_{k-1}(e')$	$N(\lvert\mathcal{E}\rvert - \lvert\mathcal{V}\rvert)$ adds
	$\alpha_k(e) = \gamma_k(e) \sum_{e':t(e')=i(e)} \alpha_{k-1}(e')$	$N\lvert\mathcal{E}\rvert$ multiplies
6.	Compute $\lambda_k(e)$	$N\lvert\mathcal{E}\rvert$ multiplies
7.	Compute $P_k(u;O)$	$N\lvert\mathcal{E}\rvert$ adds, $N\lvert\mathcal{A}\rvert$ multiplies
Total	$3N\lvert\mathcal{E}\rvert + 2N\lvert\mathcal{A}\rvert$ multiplies, $3N\lvert\mathcal{E}\rvert - 2N\lvert\mathcal{V}\rvert$ adds	

Table 7: MAP algorithm summary

For a code with $|\mathcal{E}| = |\mathcal{A}||\mathcal{V}|$, the total cost per (half)iteration is approximately

$$3N|\mathcal{E}| \text{ multiplies, } 3N|\mathcal{E}| - 2N|\mathcal{V}| \text{ adds.}$$

Compare this to the complexity of state-based BCJR, e.g., [30,31], for which the cost per (half)iteration is approximately

$$4N|\mathcal{E}| \text{ multiplies, } 3N|\mathcal{E}| - 2N|\mathcal{V}| \text{ adds.}$$

The edge-based algorithm has $\approx 25\%$ fewer multiplies per iteration. However, this advantage occurs in computing probabilities of events that are a function of the edges. As a counter-example, to compute (9) for a convolutional code where all inputs to a state have the same input label, it would be more efficient to use [31].

	$i(e)$	$t(e)$	$u(e)$	$c(e)$
$e_{;1}$	1	2	x	0000000
$e_{;2}$	2	3	x	0000000
$e_{;3}$	3	1	000	0010000
$e_{;4}$	3	1	001	0001000
$e_{;5}$	3	1	010	0000100
$e_{;6}$	3	1	011	0000010
$e_{;7}$	3	1	100	0000001
$e_{;8}$	3	4	x	0000000
$e_{;9}$	4	1	1010	0100000
$e_{;10}$	4	1	1011	0010000
$e_{;11}$	4	1	1100	0001000
$e_{;12}$	4	1	1101	0000100
$e_{;13}$	4	1	1110	0000010
$e_{;14}$	4	1	1111	0000001

Table 8: $R = 1/7$ $(d, k) = (16, 31)$ STPPM mapping

C Decoding on a Variable-Out-Degree Trellis

The encoders/decoders for the STPPM codes have a compact description via a variable-out-degree (VOD) trellis. The VOD trellis may be used as the basis for the encoding mapping. The VOD trellis may also be used without modification to form a maximum-likelihood (ML) estimate via, for example, the Viterbi Algorithm. Decisions would be delayed, however typically no longer than existing delay due to the truncation depth. On the other hand, certain modifications will be necessary to form a maximum a posteriori (MAP) estimate via, for example, the BCJR algorithm. In this section we describe these modifications, and compare the costs of using a VOD trellis to a fixed-out-degree trellis.

Assume the code is described by a time-invariant trellis (straight-forward modifications would treat time-varying trellises) 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, $i(e)$, and terminal state, $t(e)$. Edge j is denoted by $e_{;j}$, the edge at time k by e_k, and $e_{k;j}$ denotes edge j at time k. Let s_k denote the state at time k, so that $t(e_{k-1}) = s_k = i(e_k)$. Let $p_k(e|i(e)) \stackrel{\text{def}}{=} P\{e_k = e|i(e_k) = i(e)\}$ denote the conditional edge probability, $u(e)$ and $c(e)$ denote the input and output edge labels. Our running example will be the VOD trellis for a rate $1/7$ code into the $(d, k) = (16, 31)$ constraint whose input mapping is described in Table 8. Input edge labels are denoted $u_{k-2}u_{k-1}u_k$, $u_{k-3}u_{k-2}u_{k-1}u_k$, or x for 'don't care'. Output edge labels are similarly labeled such that prior bits are to the left of the current bit.

Appendix B contains a derivation of a MAP algorithm and notation in this section follows the appendix. Note that the algorithm computes probabilities of edges directly, as opposed to states, which generalizes in a straight-forward manner to a VOD trellis. Note that for a VOD trellis one cannot assume $u_k = u(e_k)$ or that $p_k(e|i(e)) = P\{u_k = u(e)\}$ since the kth input may not map to the kth edge in the trellis.

Let $\mathcal{E}^m$ denote the allowable length m paths in the trellis, i.e., $\mathcal{E}^m \stackrel{\text{def}}{=} \{e_{k:k+m-1} | i(e_j) = t(e_{j-1}), j = k, \ldots, k+m-2\}$, and $\mathcal{E}_0^m \subseteq \mathcal{E}^m$ be the subset of those paths whose input labels contain $u_k = 0$. Then

$$\begin{aligned} P_k(u=0;O) &= \sum_{e_{k:k+2} \in \mathcal{E}^3} P\{e_{k:k+2}, u_k = 0 | \mathbf{y}\} \\ &= \sum_{e_{k:k+2} \in \mathcal{E}^3} P\{e_{k:k+2} | \mathbf{y}\} P\{u_k = 0 | e_{k:k+3}, \mathbf{y}\} \\ &= \sum_{e_{k:k+2} \in \mathcal{E}_0^3} P\{e_{k:k+2} | \mathbf{y}\} \end{aligned}$$

Paths of length 3 are chosen for this trellis so that $P\{u_k = 0 | e_{k:k+3}, \mathbf{y}\}$ is an indicator function for the set $\mathcal{E}_0^3$. Note also that for this trellis $P\{e_{k:k+2} | \mathbf{y}\} = P\{e_j | \mathbf{y}\}$, where e_j is the unique edge in the path $e_{k:k+2}$ that appears in parallel to another edge. Let $\mathcal{E}_0'$ denote the collection of these edges. Then

$$P_k(u=0;O) = \sum_{e_j \in \mathcal{E}_0'} P\{e_j | \mathbf{y}\},$$

where $\mathcal{E}_0' = \{e_{k;3,5,7,9,11,13}, e_{k+1;3,4,7,11,12}, e_{k+2;3,4,5,6,9,10}\}$, and

$$P_k(u=1;O) = \sum_{e_j \in \mathcal{E}_1'} P\{e_j | \mathbf{y}\},$$

where $\mathcal{E}_1' = \{e_{k;4,6,10,12,14}, e_{k+1;5,6,9,10,13,14}, e_{k+2;7,8,11,12,13,14}\}$. Note that $e_{k+2;8}$ replaces $e_{k+3;9,10,11,12,13,14}$ in $\mathcal{E}_1'$.

It remains to specify how to solve for $p_k(e|i(e))$ given the sequence $P_k(u=0;I)$. Clearly for all edges not in parallel with another edge, $p_k(e|i(e)) = 1$. Only states 3 and 4 have parallel outgoing edges. Consider first an edge with $i(e) = 3$:

$$\begin{aligned} p_k(e_k = 3 | i(e_k) = 3) &= P\{u_k = 0, u_{k-1} = 0, u_{k-2} = 0 | i(e_k) = 3\} \\ &= P\{u_k = 0 | i(e_k) = 3\} P\{u_{k-1} = 0 | i(e_k) = 3\} P\{u_{k-2} = 0 | i(e_k) = 3\} \\ &= P_k(0;I) P_{k-1}(0;I) P_{k-2}(0;I). \end{aligned}$$

The second equation follows from an assumption that bits u_k are independent. The third equation follows since, given $i(e_k) = 3$, the prior two transitions are not data-dependent. Similar analysis yields the conditional edge probabilities for all edges with $i(e) = 3$. Consider an edge with $i(e) = 4$:

$$\begin{aligned} p_k(e_k = 11|i(e_k) = 4) &= P\{u_{k:k-3} = 0011|e_{k-1} = 8, s_{k-1} = 3\} \\ &= \frac{P\{u_{k:k-3} = 0011, e_{k-1} = 8|s_{k-1} = 3\}}{p_{k-1}(e_{k-1} = 8|i(e_{k-1}) = 3)} \\ &= \frac{P\{u_{k:k-3} = 0011|s_{k-1} = 3\}}{p_{k-1}(8|3)} \\ &= \frac{P_k(0; I)P_{k-1}(0; I)P_{k-2}(1; I)P_{k-3}(1; I)}{p_{k-1}(8|3)} \end{aligned}$$

Similar analysis yields analogous equations for all other edges with $i(e) = 4$.

We summarize the decoding operation and perform a rough analysis of the decoding complexity. Certain simplifications follow the discussion in Appendix B.

Step		Cost
1.	Receive $\mathbf{y}$	none
2.	Determine $p_k(e|i(e))$	$20N$ multiplies
3.	Compute $\gamma_k(e)$	$12N$ multiplies
4.	Compute $\beta_k(t(e))$	$14N$ multiplies
		$11N$ adds
5.	Compute $\alpha_k(e)$	$14N$ multiplies
		$11N|\mathcal{E}|$ adds
6.	Compute $\lambda_k(e)$	$14N$ multiplies
7.	Compute $P_k(u; O)$	$32N$ adds, $2N$ multiplies
Total	$76N$ multiplies, $54N$ adds	

Table 9: VOD MAP algorithm summary

The total cost per (half)iteration is

$$76N \text{ multiplications}, 54n \text{ additions}.$$

Compare this to decoding a fixed-out-degree trellis. The equivalent fixed-degree trellis has 10 states, and two outgoing edges per state. An estimate from Table 7 yields a total cost per (half)iteration of

$$64N \text{ multiplications}, 40N \text{ additions}.$$

The cost in computations per iteration favors the fixed-out-degree trellis in this example. However, we anticipate that for larger d the complexity of decoding the VOD trellis will be lower. Certain simplifications may also be made to the computations in steps 2 and 7.

References

[1] R. G. Lipes, "Pulse-position-modulation coding as near-optimum utilization of photon counting channel with bandwidth and power constraints," *DSN Progress Report*, vol. 42, pp. 108–113, Apr. 1980.

[2] A. D. Wyner, "Capacity and error exponent for the direct detection photon channel–Part I," *IEEE Transactions on Information Theory*, vol. 34, pp. 1449–1461, Nov. 1988.

[3] J. R. Pierce, "Optical channels: Practical limits with photon counting," *IEEE Transactions on Communications*, vol. COM-26, pp. 1819–1821, Dec. 1978.

[4] S. A. Butman, J. Katz, and J. R. Lesh, "Bandwidth limitations on noiseless optical channel capacity," *IEEE Transactions on Communications*, vol. COM-30, pp. 1262–1264, May 1982.

[5] H. H. Tan, "Capacity of a multimode direct detection optical communication channel," *TDA Progress Report*, vol. 42-63, pp. 51–70, May 1981.

[6] G. G. Ortiz and A. Biswas, "Design of an opto-electronic receiver for deep-space optical communications," *TMO Progress Report*, vol. 42-142, pp. 1–17, aug 2000.

[7] M. Srinivasan, J. Hamkins, B. Madden-Woods, A. Biswas, and J. Beebe, "Laboratory characterization of silicon avalanche photodiodes (APDs) for pulse-position modulation (PPM) detection," *TMO Progress Report*, vol. 42-142, pp. 1–14, Aug. 2000.

[8] K. A. S. Immink, P. H. Siegel, and J. K. Wolf, "Codes for digital recorders," *IEEE Transactions on Information Theory*, vol. 44, pp. 2260–2299, Oct. 1998.

[9] B. H. Marcus, R. M. Roth, and P. H. Siegel, "Constrained systems and coding for recording channels," in *Handbook of Coding Theory* (V. S. Pless and W. C. Huffman, eds.), ch. 20, Elsevier Science, 1998.

[10] H. H. Tan, "A statistical model of the photomultiplier gain process with applications to optical pulse detection," *TDA Progress Report*, vol. 42-68, pp. 55–67, Apr. 1982.

[11] S. Dolinar, D. Divsalar, J. Hamkins, and F. Pollara, "Capacity of PPM on Gaussian and Webb channels," *TMO Progress Report*, Aug. 2000.

[12] C. E. Shannon, "A mathematical theory of communication," *The Bell System Technical Journal*, vol. 27, pp. 379–423, July 1948.

[13] A. Khandekar and R. McEliece, "Communication over channels with dead time." preprint, July 1999.

[14] S. W. McLaughlin, "The construction of M-ary (d, ∞) codes that achieve capacity and have the fewest number of encoder states," *IEEE Transactions on Information Theory*, vol. 43, no. 2, pp. 699–703, 1997.

[15] S. W. McLaughlin, J. Luo, and Q. Xie, "On the capacity of M-ary runlength-limited codes," *IEEE Transactions on Information Theory*, vol. 41, no. 5, pp. 1508–1511, 1995.

[16] D. Slepian, "Permutation modulation," *Proceedings of the IEEE*, vol. 53, pp. 228–236, Mar. 1965.

[17] J. K. Wolf, "Permutation codes, (d, k) codes and magnetic recording," in *Proceedings 1990 IEEE Colloquium in South America, Argentina, Brazil, Chile, Uruguay* (W. Tompkins, ed.), pp. 92–102, Sept. 1990.

[18] J. Budinger, M. Vanderaar, P. Wagner, and S. Bibyk, "Combinatorial pulse position modulation for power-efficient free-space laser communications," in *Proceedings of the SPIE*, vol. 1866, (Los Angeles, CA), pp. 214–225, The International Society for Optical Engineering, Jan. 1993.

[19] S. Datta and S. W. McLaughlin, "An enumerative method for runlength-limited codes: Permutation codes," *IEEE Transactions on Information Theory*, vol. 45, pp. 2199–2204, Sept. 1999.

[20] P. A. Franaszek, "Sequence-state coding for digital transmission," *The Bell System Technical Journal*, vol. 47, pp. 143–157, 1968.

[21] K. A. S. Immink, *Codes for Mass Data Storage Systems*. The Netherlands: Shannon Foundation Publishers, 1999.

[22] S. Dolinar, "Pulsed optical communication with a dead-time constraint." preprint, Feb. 2001.

[23] B. H. Marcus, P. H. Siegel, and J. K. Wolf, "Finite-state modulation codes for data storage," *IEEE Journal Selected Areas Communications*, vol. 10, pp. 5–37, Jan. 1992.

[24] B. H. Marcus and R. M. Roth, "Bounds on the number of states in encoder graphs for input-constrained channels," *IEEE Transactions on Information Theory*, vol. 37, pp. 742–758, May 1991.

[25] S. Benedetto, D. Divsalar, G. Montorsi, and F.Pollara, "A soft-input soft-output maximum a posteriori (MAP) module to decode parallel and serial concatenated codes," *TDA Progress Report*, vol. 42-127, pp. 1–20, 1996.

[26] J. L. Fan, "Constrained coding techniques for soft iterative decoding." preprint, Aug. 1999.

[27] K. A. S. Immink, "A practical method for approaching the channel capacity of constrained channels," *IEEE Transactions on Information Theory*, vol. 43, pp. 1389–1399, Sept. 1997.

[28] J. Hamkins, "Performance of binary turbo coded 256-ppm," *TMO Progess Report*, vol. 42, pp. 1–15, Aug. 1999.

[29] 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.

[30] K. Adrews, "Ken's turbo decoder." preprint, 2001.

[31] 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.