

# RATE COMPATIBLE PUNCTURED TURBO (RCPT) CODES IN A HYBRID FEC/ARQ SYSTEM \*

Douglas N. Rowitch and Laurence B. Milstein  
University of California, San Diego

## Abstract

*This paper introduces a Hybrid FEC/ARQ system that employs Rate Compatible Punctured Turbo (RCPT) codes to achieve enhanced throughput performance over a nonstationary Gaussian channel. The proposed RCPT-ARQ system combines the performance of turbo codes with the frugal use of incremental redundancy inherent in the rate compatible punctured convolutional (RCPC) codes of Hagenauer. Moreover, this paper introduces the notion of puncturing the systematic code symbols of a turbo code to maximize throughput at signal-to-noise ratios (SNR) of interest. The resulting system provides both an efficient family of achievable code rates at middle to high SNR, and powerful low rate error correction capability at low SNR.*

## 1 Introduction

Conventional applications of forward error correction (FEC) coding usually consist of the selection of fixed rate coding schemes which are well suited to the channel characteristics and the acceptable error rates for the data to be transmitted [1]. In some cases, however, a more flexible scheme may be desirable to accommodate different error protection requirements, or channels with unknown or time varying parameters [1]. In such cases, automatic repeat-request (ARQ) protocols have been successfully employed to adapt to changes in the transmission medium. In particular, hybrid FEC/ARQ protocols have been established which exploit both the predictable performance of FEC codes and the rate flexibility of ARQ protocols [1], [2]. Hagenauer [2] introduced rate compatible punctured convolutional (RCPC) codes with such an application in mind.

RCPC codes are constructed from a single rate  $1/M$  convolutional code, wherein a family of higher rate codes are formulated by puncturing successively greater numbers of code symbols. These codes have practical utility in that the system requires a single rate  $1/M$  convolutional encoder and Viterbi decoder. The transmitter and receiver need only share a puncturing table to determine which code symbols to transmit at a given time, and the receiver may simply insert erasures for all code symbols that have not yet been received. The rate compatibility requirement of these codes restricts the puncturing rule such that all of the code symbols of a high rate punctured code are used by the lower rate codes, i.e., the high rate codes are embedded in the lower rate codes. In this manner, the transmitter need only

transmit *supplemental* code symbols to get to the next lowest rate code. RCPC-ARQ protocols, therefore, fall into the so-called class of *incremental redundancy* codes, in that check digits are incrementally transmitted to adaptively meet the error performance requirements of the system. These codes are also appealing from an information theoretic standpoint, since no received information is discarded (as in other hybrid FEC/ARQ schemes [1]), especially when used in concert with Chase's code combining [3].

Turbo codes, introduced by Berrou, *et al.* in 1993 [4], caused a great stir in the coding community and have prompted a great deal of research. We restrict our attention to interleaved, parallel concatenated codes (PCC), since they are directly applicable to the partial transmission scheme to be proposed in this paper and, in particular, we consider recursive systematic convolutional (RSC) constituent codes since they tend to exhibit the best performance [4]. The turbo encoder considered in this paper consists of the composite of two or more RSC encoders, each separated by a pseudo-random interleaver of block size  $N$ , resulting in a composite encoder of overall rate  $1/M$ . Transmitted are the systematic (i.e., information) code symbols of the first encoder, and the non-systematic (i.e., parity) code symbols of all constituent encoders. The receiver provides a soft-output maximum *a posteriori* (MAP) decoder corresponding to each RSC encoder.

Rate compatible turbo codes have been considered to achieve unequal error protection [5], and the use of turbo codes in an ARQ protocol has been proposed [6]; however, this is the first paper known to the authors which considers rate compatible punctured turbo (RCPT) codes as applied within a high throughput ARQ strategy. The formulation of the RCPT codes follows that of the RCPC codes almost directly; however, there are several notable distinctions. The most obvious distinction is that the rate  $1/M$  convolutional encoder is replaced by a rate  $1/M$  turbo encoder and, of course, the rate  $1/M$  soft decision Viterbi decoder is replaced with a bank of MAP decoders and the associated iterative decoding structure. The more subtle distinction considered in this paper has to do with the process for selecting the puncturing tables corresponding to different transmission rates of the protocol. For RCPC codes, the puncturing tables were selected such that each successive transmission of parity symbols yielded the greatest possible increment in the free distance of the new lower rate code. For RCPT codes, however, there are other, perhaps more important, criteria. In particular, the subsets of code symbols that have been

---

\*This work is supported in part by the National Science Foundation under Grant No. NCR-9213140 and the Focused Research Initiative on Wireless Multimedia Networks under Grant No. DAAH04-95-1-0248.

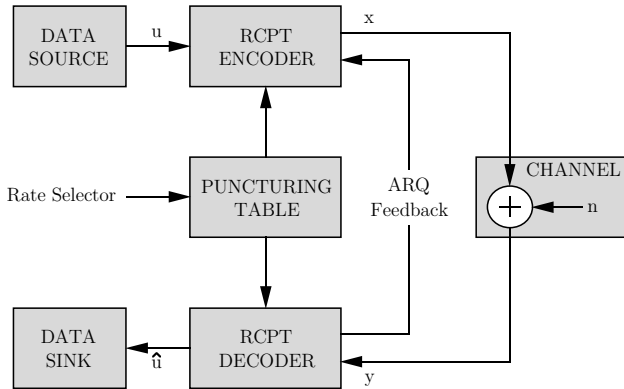


Figure 1: Data transmission using RCPT codes over the Gaussian channel.

transmitted at a given stage implicitly determine the number of MAP decoders that can be employed in the receiver. Indeed, if non-systematic parity symbols have not been received for two or more RSC encoders, iterative decoding is not possible. It turns out that there are situations where it is profitable, at high code rates, to puncture some of the systematic code symbols so that more non-systematic symbols may be sent, and thus utilize more or all of the MAP decoders resident in the receiver.

In Section 2, we present an overview of the RCPT-ARQ system, and in Section 3, we conduct an analysis of system performance. Section 4 contains numerical results and, finally, in Section 5 can be found the conclusions.

## 2 FEC/ARQ System Overview

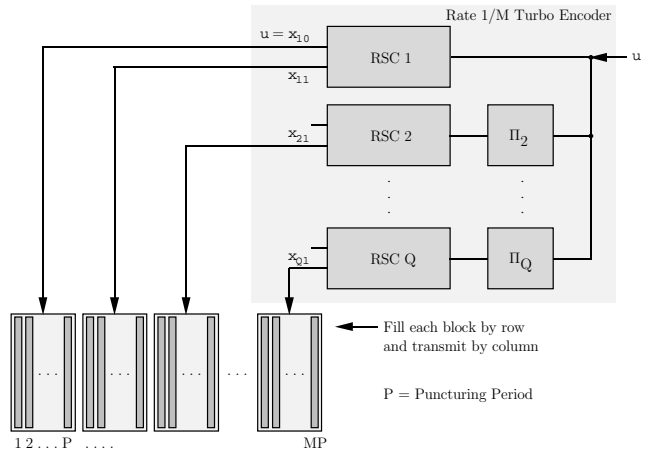
### 2.1 Channel Assumptions

We assume coherent binary signalling over an additive white Gaussian channel at a given symbol signal-to-noise ratio,  $E_s/N_0$ . We may, therefore, characterize the output of the channel, at any given time, as  $y = x + n$ , where  $x \in \{+1, -1\}$  and  $n$  is a zero-mean Gaussian random variable with standard deviation,  $\sigma = \sqrt{N_0/2E_s}$ . Fig. 1 depicts the RCPT-ARQ system, where the rate selector determines which subset of systematic and non-systematic parity symbols to transmit during a given signalling interval.

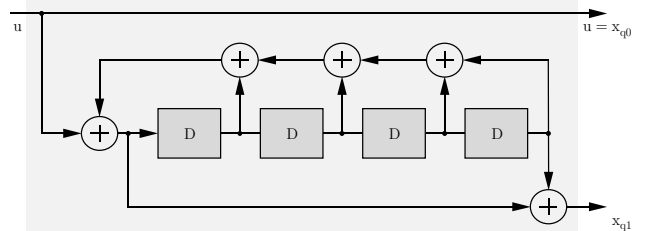
We, furthermore, assume an errorless, low capacity feedback channel over which the receiving system can transmit positive (ACK) or negative (NAK) acknowledgements.

### 2.2 RCPT Codes

The binary data source is encoded into blocks in accordance with a low redundancy  $(N, K)$  block error detection code. This encoded data source is input to the RCPT encoder which (a) turbo encodes the data sequence and (b) partitions the resulting code symbols in sub-blocks of  $(N/P)$  symbols, where  $P$  is the so-called puncturing period. In Fig. 2a, an example RCPT encoder is formed from an underlying rate  $1/M$  turbo encoder, consisting of  $Q$  rate  $1/2$  RSC encoders. Note that the composite encoder can, in general, be formed by using a fewer number of lower rate RSC encoders, and that the constituent RSC code rates need not



a. Example RCPT encoder.



b. Example RSC encoder.

Figure 2: Example RCPT-ARQ encoder implementation.

be equal. For example, if  $M = 4$ , then one could use either three rate  $1/2$  RSC encoders ( $Q = 3$ ), or one rate  $1/3$  and one rate  $1/2$  RSC encoder ( $Q = 2$ ). Since the turbo code is of rate  $1/M$ , there are  $MP$  total sub-blocks produced for potential transmission. It is seen in the figure that the systematic bits of all but the first encoder are discarded and that the resulting  $M$  parity streams are represented as matrices with  $P$  columns and  $(N/P)$  bits per column (each column is a transmission sub-block). Each parity stream fills its matrix by row and is transmitted by column (like a block interleaver)<sup>1</sup>. Note that tail-bits can be optionally transmitted for one or more of the constituent encoders, in which case we use the trellis termination scheme in [7]; alternatively, we can avoid the transmission of tail-bits entirely [8]. The trade-off here is reduced throughput due to tail-bit transmission (especially severe for short block sizes) versus degradation in MAP decoder performance and/or increased delay in iterative decoding. The interleavers,  $\pi_q$ ,  $q = 2 \dots Q$ , are taken to be ‘‘S-random’’ permutations [7], as they tend to provide superior performance for the short frame sizes typically used in ARQ protocols. S-random permutations satisfy a distance condition ensuring that each symbol to be permuted is a distance  $S$  or more from the previous  $S$  adjacent symbols. Fig. 2b illustrates an example rate  $1/2$  constituent RSC encoder of memory  $\nu = 4$ , and generator matrix

<sup>1</sup>The matrices are used to illustrate the puncturing operation and need not be used in implementation.

$(1, g_b/g_a)$ , where the generator polynomials  $g_a$  and  $g_b$  have octal representations  $(37)_{octal}$  and  $(21)_{octal}$ , respectively.

The RCPT puncturing rule amounts to sending collections of one or more columns of the parity matrices, such that at least  $P$  columns are sent in the initial transmission and no column is sent twice. The code construction allows for a family of codes of rates

$$R_\ell = \frac{P}{P + \ell}, \quad \ell = 0, 1, \dots, (M - 1)P. \quad (1)$$

For each value of  $\ell$ , we define a binary  $M \times P$  puncturing matrix,  $\mathbf{a}(\ell)$ . If  $a_{ij}(\ell) = 1$ , then the  $j^{\text{th}}$  column of the  $i^{\text{th}}$  parity matrix belongs to the sub-code of rate  $R_\ell$ . Therefore, based on the above restrictions,  $\mathbf{a}(0)$  must contain  $P$  or more ones,  $\mathbf{a}(\ell + 1)$  must have ones in the same positions as in  $\mathbf{a}(\ell)$  plus an additional one, and, finally,  $\mathbf{a}((M - 1)P)$  is all ones.

As an example, suppose we construct an RCPT code with  $M = 3$  and  $P = 4$  formed from two rate  $1/2$  constituent RSC encoders. Then, the RCPT code can select any code rate in the set  $\{1, \frac{4}{5}, \frac{2}{3}, \frac{4}{7}, \frac{1}{2}, \frac{4}{9}, \frac{2}{5}, \frac{4}{11}, \frac{1}{3}\}$ , by selecting, respectively, 4, 5, ..., 12 columns from the parity matrices. The set of puncturing matrices could, for example, take the form

$$\begin{array}{cccc} \mathbf{a}(0) & \mathbf{a}(1) & \mathbf{a}(2) & \mathbf{a}(8) \\ \left[ \begin{array}{c} 1111 \\ 0000 \\ 0000 \end{array} \right] & \left[ \begin{array}{c} 1111 \\ 0010 \\ 0000 \end{array} \right] & \left[ \begin{array}{c} 1111 \\ 0010 \\ 1000 \end{array} \right] & \dots \left[ \begin{array}{c} 1111 \\ 1111 \\ 1111 \end{array} \right] \end{array}$$

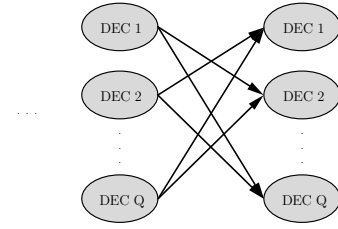
Note that, for this example, all four columns of systematic bits (first row) are sent in the first packet ( $\mathbf{a}(\ell = 0)$ ), the third column of the first encoder's non-systematic parity bits are sent in the second packet ( $\mathbf{a}(\ell = 1)$ ), and the first column of the second encoder's non-systematic parity bits are sent in the third packet ( $\mathbf{a}(\ell = 2)$ ). It is only after this third packet that the receiver can invoke the iterative decoder. In implementation, one need not utilize all possible code rates. Indeed, for ARQ applications with large channel variations, an exponential increase in  $\ell$  may be desirable (e.g.,  $\ell = 1, 2, 4, 8, \dots$ ) [2]. It should be clear at this point that the transmission packet size will, in general, be variable.

### 2.3 RCPT-ARQ Protocol

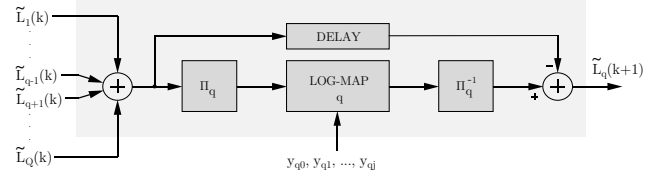
We assume, without loss of generality, a selective-repeat ARQ strategy due to its bandwidth efficiency, noting that the RCPT-ARQ protocol can be trivially adapted to either stop-and-wait or go-back-N schemes. Furthermore, we do not consider the effects of finite memory or unreliable feedback on the performance of the system.

The RCPT-ARQ protocol performs the following steps:

1. Encode  $K$  information digits in accordance with the  $(N, K)$  block error detection code.
2. Input the  $N$  bit coded information block into the rate  $1/M$  RCPT encoder. Store the resulting  $M$  parity streams at the transmitter, possibly in matrices as depicted in Fig. 2a, for potential transmission.
3. Initialize  $\ell$  (e.g., set  $\ell = 1$ ).



a. Parallel decoder structure.



b. Structure of the  $q^{\text{th}}$  decoder.

Figure 3: Decoder implementation.

4. Transmit the columns as indicated by  $\mathbf{a}(\ell)$  that have not yet been transmitted.
5. Attempt to decode the rate  $R_\ell$  code using the code symbols received thus far, inserting erasures for all those symbols not yet received. Within the iterative decoding loop after each iterative update:
  - (a) Hard quantize the cumulative vector of likelihood ratios on the information sequence.
  - (b) Calculate the syndrome of the  $(N, K)$  codeword.
  - (c) If there is an all-zero syndrome, exit the loop, output the  $K$  decoded information digits and send an ACK to the transmitter. Otherwise, continue the decoding loop until a prescribed maximum number of iterations, upon which, if the syndrome is still non-zero, exit the loop and send a NAK to the transmitter.
6. At the transmitter, if an ACK is received, reset the protocol and proceed to step 1.
7. Otherwise, if a NAK is received, increment  $\ell$  to the next appropriate value and proceed to step 4.

If decoding is not successful after the transmission of all columns of code symbols (i.e.,  $\ell = (M - 1)P$  and  $R_\ell = 1/M$ ), then the protocol resets, commencing with step 3. The receiver, in turn, may optionally reset its receive buffers, or, ideally, retain the previously received quantities and employ code combining [3] to merge the data.

Note that steps 5a through 5c can be performed directly on the received values corresponding to the systematic code symbols (assuming all  $N$  symbols have been received), and potentially avoid the overhead of the iterative decoding process.

### 2.4 RCPT Decoder

The decoder for the RCPT code is not unlike the decoder for conventional turbo codes. We use the parallel decoder

structure as depicted in Fig. 3a, which illustrates the iterative decoding process, wherein the outputs of each decoder feed all *other* decoders [7]. Of course, if no parity information has been received corresponding to the  $q^{\text{th}}$  decoding element, then the  $q^{\text{th}}$  decoding element will be excluded from the iterative process. The  $q^{\text{th}}$  decoding element is shown in Fig. 3b. Each decoding element contains a MAP decoder that produces a soft vector of so-called *extrinsic* likelihood ratios, which reflect the incremental contribution of the decoder to the overall likelihood of the information sequence (in rough terms: the output likelihood minus the input likelihood). The extrinsic likelihood vectors output from all other decoders at time index  $k$ ,  $\tilde{\mathbf{L}}_i(k)$ ,  $i = 1 \dots Q$ ,  $i \neq q$ , are summed and used as input to the  $q^{\text{th}}$  decoding element<sup>2</sup>. This input vector is first interleaved using the pseudo-random interleaver,  $\pi_q$ . In this manner, the likelihoods are ordered consistently with the parity information of the  $q^{\text{th}}$  encoder. The permuted likelihood data is used in conjunction with the received systematic code vector,  $\mathbf{y}_{q0}$ , and non-systematic code vector(s),  $\mathbf{y}_{q1}, \dots, \mathbf{y}_{qj_q}$ , to produce a new estimate of the likelihood ratios of the  $N$  information symbols. Note that  $\mathbf{y}_{q0}$  is produced by permuting  $\mathbf{y}_{10}$ ; i.e.,  $\mathbf{y}_{q0} = \pi_q [\mathbf{y}_{10}]$ . By subtracting the input likelihood vector from the deinterleaved output likelihood vector, we obtain the extrinsic likelihood vector<sup>3</sup>.

The MAP algorithm, in this context, was originally derived by Bahl *et al.*[9]. We refer the reader to [10] for a comprehensive treatment on the implementation of the MAP decoder for use within a turbo decoder. For the simulations used in this paper, we used a LOG-MAP implementation which produces log-likelihood ratios for the information sequence.

### 3 Throughput Analysis

Let a frame denote the binary vector of  $K$  digits which are produced by the decoder. We use primarily average throughput ( $R_{AV}$ ) and also the probability of a frame error (FER) as the figures of merit for the RCPT-ARQ system. Combining the rate of the block error detection code and (1), we have<sup>4</sup>

$$R_{AV} = \frac{K}{N} \cdot \frac{P}{P + \ell_{AV}}, \quad (2)$$

where  $\ell_{AV}$  is the average number of additionally transmitted columns. We assume that the number of parity bits,  $N - K$ , used in the error detecting code, are sufficient to guarantee a negligible probability of undetected frame error.

Suppose that an implementation of the RCPT-ARQ protocol selects puncturing tables via the indices,  $\ell_1 < \ell_2 < \dots < \ell_k < \dots < \ell_K$ , and denote  $P_F(\ell_k)$  as the probability that the FEC decoding attempt at the  $k^{\text{th}}$  step results in

<sup>2</sup>Prior to the first iteration, this vector is initialized to zero (equally likely).

<sup>3</sup>Note that in the LOG-MAP implementation of the MAP algorithm, the extrinsic likelihoods can be calculated directly, rendering the delay element and subtraction unnecessary.

<sup>4</sup>If tail-bits are transmitted for one or more of the constituent encoders, the average throughput will be reduced accordingly.

errors that are detected by the  $(N, K)$  error detection code. Then, assuming that the protocol terminates (fails) after  $K$  transmissions, the frame error rate is simply<sup>5</sup>

$$FER = P_F(\ell_K). \quad (3)$$

If we additionally assume that the  $K$  decoding attempts have statistically independent outcomes, we have [2]

$$\ell_{AV} = \sum_{k=1}^K \ell_k (1 - P_F(\ell_k)) \cdot \prod_{i=0}^{k-1} P_F(\ell_i) + \ell_K \prod_{i=0}^K P_F(\ell_i), \quad (4)$$

where we define  $P_F(\ell_0) = 1$ .

Given an upper bound on  $P_F(\ell_k)$ , which we denote by  $P_{UB}(\ell_k)$ , the FER can be trivially upper bounded as

$$FER < P_{UB}(\ell_K), \quad (5)$$

and the average throughput can be lower bounded by substituting  $P_{UB}(\ell_k)$  for  $P_F(\ell_k)$  in (4) and substituting the result in (2). While initial results presented in this paper are based on simulations, it would be desirable to apply transfer function bounding techniques [11] to obtain  $P_{UB}(\ell_k)$  to aid the search for optimal puncturing matrices, for a given encoder.

### 4 Numerical Results

In this section, we present preliminary numerical results of simulations of the RCPT-ARQ protocol. Plotted in Fig. 4 are the average rates attained by several variants of the RCPT-ARQ algorithm. We used a RCPT encoder made up of two RSC encoders, as depicted in Fig. 2b with  $P = 8$ , and show throughput for two selections of puncturing tables. In the first case, we use the more conventional RCPC approach of sending the  $P$  columns of systematic code symbols ( $\ell = 0$ ) and rely on the error detection code alone at the receiver. If a retransmission is required, we must decrease the rate to  $4/5$  ( $\ell = 2$ ) such that two columns of non-systematic parity symbols may be sent, making an iterative decoding attempt possible. In the second case, we initially puncture one of the 8 columns of systematic bits and send two columns of non-systematic bits, yielding a punctured rate of  $8/9$  ( $\ell = 1$ ). The specific puncturing tables used (in octal) are, for case 1,

$$\begin{array}{cccccccc} \mathbf{a}(0) & \mathbf{a}(2) & \mathbf{a}(4) & \mathbf{a}(6) & \mathbf{a}(8) & \mathbf{a}(10) & \mathbf{a}(12) & \mathbf{a}(14) & \mathbf{a}(16) \\ \left[ \begin{array}{c} 377 \\ 000 \\ 000 \end{array} \right] & \left[ \begin{array}{c} 377 \\ 002 \\ 001 \end{array} \right] & \left[ \begin{array}{c} 377 \\ 102 \\ 201 \end{array} \right] & \left[ \begin{array}{c} 377 \\ 112 \\ 221 \end{array} \right] & \left[ \begin{array}{c} 377 \\ 312 \\ 321 \end{array} \right] & \left[ \begin{array}{c} 377 \\ 352 \\ 325 \end{array} \right] & \left[ \begin{array}{c} 377 \\ 353 \\ 365 \end{array} \right] & \left[ \begin{array}{c} 377 \\ 357 \\ 375 \end{array} \right] & \left[ \begin{array}{c} 377 \\ 377 \\ 377 \end{array} \right], \end{array}$$

and, for case 2, we replace  $\mathbf{a}(0)$  with  $\mathbf{a}(1)$ , where

$$\mathbf{a}(1) = \left[ \begin{array}{c} 376 \\ 002 \\ 001 \end{array} \right].$$

For these results, we allowed the protocol to repeat three additional times and used code combining to merge repeated

<sup>5</sup>The FER will be reduced if the protocol is allowed to repeat and code combining is employed.

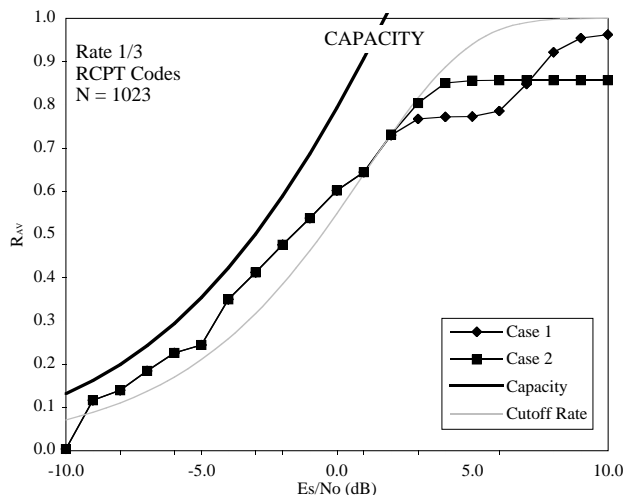


Figure 4: Throughput of two RCPT-ARQ codes over the Gaussian channel. Codes:  $P = 8$ ,  $M = 3$ ,  $Q = 2$ ,  $\nu = 4$ ,  $N = 1023$ ,  $K = 993$ , and  $S = 20$ .

data. It is immediately seen that by puncturing a small percentage of the systematic code symbols (case 2), one is able to sustain higher throughput at the lower signal-to-noise ratios of interest. These curves are shown as compared to the capacity ( $C$ ) and the computational cutoff rate ( $R_0$ ) for the discrete additive white Gaussian channel with an average power constraint, which are given, respectively, by

$$C = \frac{1}{2} \log_2 \left( 1 + \frac{2E_s}{N_0} \right) \quad \text{and} \quad (6)$$

$$R_0 = 1 - \log_2 \left( 1 + e^{-E_s/N_0} \right). \quad (7)$$

It is clear that the RCPT-ARQ codes exhibit the exceptional performance of turbo codes at very low SNRs, noting that the simulated throughput exceeds the cutoff rate for SNR below 1 (dB) and provides for error free communication at 1 – 2 (dB) from Shannon capacity.

In Fig. 5, we compare the RCPT-ARQ system from Fig. 4 (case 2) to the rate-1/3, 64-state RCPC-ARQ system of Hagenauer [2]. One notes that the RCPT system outperforms the RCPC system by as much as 2 (dB). In implementation, the RCPT system will probably be of greater complexity and be subject to greater decoding delay, and thus we see a tradeoff between throughput and complexity/delay.

## 5 Conclusion

In this paper, we introduced a novel application of turbo codes to a hybrid FEC/ARQ system. RCPT codes provide a versatile family of achievable code rates, and may be processed with a single encoder and decoder. These codes were shown to perform near the Shannon limit at low SNR, while providing excellent throughput when the SNR is high. This paper also introduced the concept of puncturing the *systematic* code symbols of a turbo code to obtain better performance at high code rates. Future research will investigate

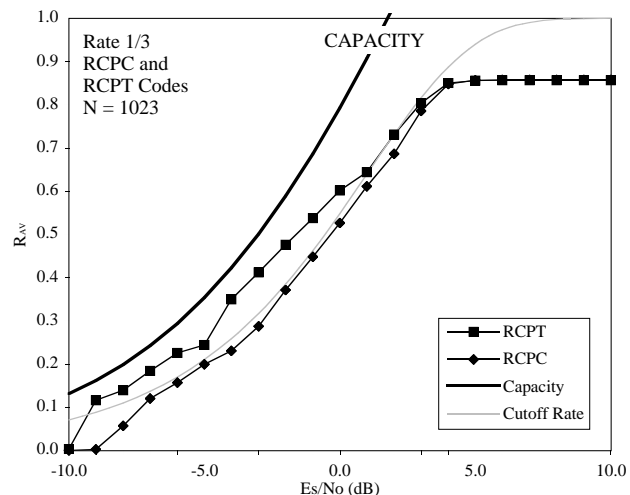


Figure 5: Throughput comparison between RCPT-ARQ and RCPC-ARQ codes over the Gaussian channel. Codes:  $P = 8$ ,  $M = 3$ ,  $Q = 2$ ,  $\nu = 4/6$  (RCPT/RCPC),  $N = 1023$ ,  $K = 993$ , and  $S = 20$ .

the performance of these codes in multipath fading environments, and throughput maximizing techniques for the selection of turbo codes and associated puncturing tables.

## References

- [1] S. Lin and D. J. Costello, Jr., *Error Control Coding: Fundamentals and Applications*. Englewood Cliffs, New Jersey: Prentice Hall, 1984.
- [2] J. Hagenauer, "Rate Compatible Punctured Convolutional Codes (RCPC Codes) and their Applications," *IEEE Trans. Commun.*, Vol. 36, Apr. 1988, pp. 389-400.
- [3] D. Chase, "Code Combining – A Maximum Likelihood Decoding Approach for Combining an Arbitrary Number of Noisy Packets," *IEEE Trans. Commun.*, Vol. 33, May. 1985, pp. 385-393.
- [4] C. Berrou, A. Glavieux and P. Thitimajshima, "Near Shannon Limit Error-Correcting Coding and Decoding: Turbo Codes," in *Proc. ICC'93*, Geneva, Switzerland, May. 1993, pp. 1064-1070.
- [5] A. S. Barbulescu and S. S. Pietrobon, "Rate Compatible Turbo Codes," *Electron. Lett.*, Vol. 31, Mar. 1995, pp. 535-536.
- [6] K. R. Narayanan and G. L. Stuber, "A Novel ARQ Technique using the Turbo Coding Principle," *Commun. Lett.*, Vol. 1, Mar. 1997, pp. 49-51.
- [7] D. Divsalar and F. Pollara, "Turbo Codes for PCS Applications," in *Proc. ICC'95*, Seattle, WA, Jun. 1995, pp. 54-59.
- [8] M. C. Reed and S. S. Pietrobon, "Turbo-Code Termination Schemes and a Novel Alternative for Short Frames," in *Proc. PIMRC'96*, Taipei, Taiwan, Oct. 1996, pp. 354-358.
- [9] L. R. Bahl, J. Cocke, F. Jelinek and J. Raviv "Optimal Decoding of Linear Codes for Minimizing Symbol Error Rate," *IEEE Trans. Info. Thy.*, Vol. 20, Mar. 1974, pp. 284-287.
- [10] S. Benedetto, G. Montorsi, D. Divsalar and F. Pollara, "Soft-Output Decoding Algorithms in Iterative Decoding of Turbo Codes," JPL TDA Progress Report 42-124, Feb. 1996, pp. 63-87.
- [11] D. Divsalar, S. Dolinar, R. J. McEliece and F. Pollara, "Transfer Function Bounds on the Performance of Turbo Codes," JPL TDA Progress Report 42-122, Aug. 1995, pp. 44-55.