A Joint Source/Channel Coder Design*

Khalid Sayood
Dept. of Electrical Engineering
and the
Center for Communication& Information Science
University of Nebraska
Lincoln, NE 68588-0511
Telephone: (402) 472-6688
FAX: (402) 472-4732
email: ksayood@eecomm.unl.edu

Fuling Liu
Western Atlas Company
Houston, Texas

Jerry D. Gibson
Dept. of Electrical Engineering
Texas A&M University
College Station, TX 77843

COMSOC Technical Committee: Signal Processing and Communication Electronics
Hot topic session number: HT28

Abstract

We examine the situation where there is residual redundancy at the source coder output. We have previously shown that this residual redundancy can be used to provide error correction without a channel encoder. In this paper we extend this approach to conventional source coder/convolutional coder combinations. We also develop a design for nonbinary encoders for this situation. We show through simulation results that the proposed systems consistently outperform conventional source-channel coder pairs with gains of greater than 10dB at high probability of error.

*This work was supported in part by NASA Lewis Research Center (NAG 3-806) and NASA Goddard Space Flight Center (NAG 5-916)

1A.

1 Introduction

One of Shannon's many fundamental contributions was his result that source coding and channel coding can be treated separately without any loss of performance for the overall system [1]. The basic design procedure is to select a source encoder which changes the source sequence into a series of independent, equally likely binary digits followed by a channel encoder which accepts binary digits and puts them into a form suitable for reliable transmission over the channel. However, the separation argument no longer holds if either of the following two situations occur:

i. The input to the source decoder is different from the output of the source encoder, which happens when the link between the source encoder and source decoder is no longer error free, or

ii. The source coder output contains redundancy.

Case (i) occurs when the channel coder does not achieve zero error probability and case (ii) occurs when the source encoder is suboptimal. These two situations are common occurrences in practical systems where source or channel models are imperfectly known, complexity is a serious issue, or significant delay is not tolerable. Approaches developed for such situations are usually grouped under the general heading of joint source/channel coding.

Most joint source channel coding approaches can be classified in two main categories; (A) approaches which entail the modification of the source coder/decoder structure to reduce the effect of channel errors, [2-18] and (B) approaches which examine the distribution of bits between the source and channel coders [19-21]. The first set of approaches can be divided still further into two classes. One class of approaches examines the modification of the overall structure [2-10], while the other deals with the modification of the decoding procedure to take advantage of the redundancy in the source coder output.

In this paper we present an approach to joint source/channel coder design, which belongs to category A, and hence we explore a technique for designing joint source/channel coders, rather than ways of distributing bits between source coders and channel coders. We assume that the two nonideal situations referred to earlier are present. For a nonideal source coder, we use MAP arguments to design a decoder which takes advantage of redundancy in the source coder output to perform error correction. We have previously shown that this approach can provide error protection at high error rates [16, 17]. In this paper we show that the use of such a decoder in conjunction with a channel encoder can provide excellent error protection over a wide range of channel error probabilities. We then use the decoder structure to infer a channel encoder structure which is similar to a nonbinary convolutional encoder.

2 The Design Criterion

For a discrete memoryless channel (DMC), let the channel input alphabet be denoted by $A = \{a_0, a_1, \ldots, a_{M-1},\}$, and the channel input and output sequences by $Y = \{y_0, y_1, \ldots, y_{L-1}\}$ and $\hat{Y} = \{\hat{y}_0, \hat{y}_1, \ldots, \hat{y}_{L-1}\}$, respectively. If $\mathcal{A} = \{A_i\}$ is the set of sequences $A_i = \{\alpha_{i,0}, \alpha_{i,1}, \ldots, \alpha_{i,L-1}\}$, $\alpha_{i,k} \epsilon A$, then the optimum receiver (in the sense of maximizing the probability of making a correct decision) maximizes $P[C]$, where

$$P[C] = \sum_{A_i} P[C|\hat{Y}]P[\hat{Y}].$$

This in turn implies that the optimum receiver maximizes $P[C|\hat{Y}]$. When the receiver selects the output to be A_k, then $P[C|\hat{Y}] = P[Y = A_k|\hat{Y}]$. Thus, the optimum receiver selects the sequence A_k such that

$$P[Y = A_k|\hat{Y}] \geq P[Y = A_i|\hat{Y}] \ \forall_i.$$

Noting that

$$P(Y|\hat{Y} = \frac{P(\hat{Y}|Y)P(Y)}{P(\hat{Y})}$$

and for fixed length codes $P(\hat{Y})$ is irrelevant to the receiver's operation, the optimal receiver maximizes $P(\hat{Y}|Y)P(Y)$. If we impose a first order markov assumption on $\{y_i\}$, we can easily show that

$$P\left(\hat{Y}|Y\right) P\left(Y\right) = \prod P\left(\hat{y}_i|y_i\right) P\left(y_i|y_{i-1}\right)$$

This result addresses the situation in case (ii), i.e., the situation in which the source coder output (which is also the channel input sequence) contains redundancy. Using this result, we can design a decoder which will take advantage of dependence in the channel input sequence. The physical structure of the decoder can be easily obtained by examining the quantity to be maximized. The optimum decoder maximizes $P(\hat{Y}|Y)P(Y)$ or equivalently $\log P(\hat{Y}|Y)P(Y)$, but

$$\log P(\hat{Y}|Y)P(Y) = \sum \log P(\hat{y}_i|y_i)P(y_i|y_{i-1}) \tag{1}$$

which is similar in form to the path metric of a convolutional decoder. Error correction using convolutional codes is made possible by explicitly limiting the possible codeword to codeword transitions, based on the previous code input and the coder structure. At the receiver the decoder compares the received data stream to the *a priori* information about the code structure. The output of the decoder is the sequence that is most likely to be the transmitted sequence. In the case where there is residual strucure in the source coder output, the structure makes some sequences more likely to be the transmitted sequence, given a particular received sequence. In other words, even when there is no structure being imposed by the encoder, there is sufficient residual structure in the source coder output that can be used for error correction. The structure is reflected in the conditional probabilities, and can be utilized via the path metric in (1) in a decoder similar in

structure to a convolutional decoder. However, to implement this decoder we need to be able to compute the path metric.

Examining the branch metric, we see that it consists of two terms $\log P(\hat{y}_i|y_i)$ and $\log P(y_i|y_{i-1})$. The first term depends strictly on our knowledge of the channel. The second term depends only on the statistics of the source sequence. In our simulation results we have assumed that the channel is a binary symmetric channel with known probability of error. We have obtained the second term using a training sequence.

In [17] we showed that the use of the decoder led to dramatic improvements under high error rate conditions. However at low error rates the performance improvement was from nonexistent to minimal. This is in contrast to standard error correcting approaches, in which the greatest performance improvements are at low error rates, with a rapid deterioration in performance at high error rates. In this work we combine the two approaches to develop a joint source channel codec which provides protection equal to the standard channel encoders at low error rates while also providing significant error protection at high error rates.

3 Convolutional Encoders and Joint Source/Channel Decoder

As convolutional coders provide excellent error protection at low error rates, and have a decoder structure similar to the JSC decoder, one way we can combine the two approaches is to obtain the transition probabilities of the convolutional encoder output and use the Joint Source/Channel (JSC) decoder described above instead of the conventional convolutional decoder.

The convolutional decoder uses the structure imposed by the encoder and the Hamming metric to provide error protection. The decoder does not use any of the residual structure from the source coder output. We can make use of the residual structure by noting that the path labels transmitted by the convolutional encoder comprise the channel input alphabet $\{y_i\}$. We can then use a training sequence to obtain the transition probabilities $P(y_i|y_{i-1})$, and an estimate of the channel error

probability to obtain $P(\hat{y}_i|y_i)$. These can be used to compute the branch metric L which can be used instead of the Hamming metric in the decoder.

We simulated this approach using a two bit DPCM system as the source encoder. We used the two images shown in Figure 1 as the source. The USC Girl image was used for training (obtaining the requisite transition probabilities) and the USC Couple image for testing. The output of the DPCM system was encoded using a (2,1,3) convolutional encoder with connection vectors

$$g^{(1)} = 64 \quad g^{(2)} = 74.$$

The convolutional encoder was obtained from [23]. The performance of the different systems was evaluated using two different measures. One was the reconstruction signal-to-noise ratio (RSNR) defined as

$$RSNR = 10 \log_{10} \frac{\sum u_i^2}{\sum (u_i - \hat{u}_i)^2}$$

where u_i is the input to the source coder (source image) and $\hat{u}_i$ is the output of the source decoder (reconstructed image). The other performance measure was the decoded error probability. The received sequence was decoded using either a standard convolutional decoder or the JSC decoder. A block diagram of the system is shown in Figure 2. The results are presented in Figure 3. While there is some improvement in the decoded error probability for high error rates, the RSNR actually goes down for the MAP decoded sequence. This is somewhat disappointing until one realizes that the JSC decoder makes use of the structure in the nonbinary output of the source coder. When we used the (2,1,3) coder we destroyed some of this structure because the source coder puts out two bit words while the channel coder codes the input one bit at a time. Therefore, if we could preserve the structure in the source coder output by coding the two bit words as a unit we should get improved performance. To verify this we conducted another set of simulation with a rate 1/2

(4,2,1) convolutional code with connection vectors

$$g_1^{(1)} = 6 \quad g_1^{(2)} = 0 \quad g_1^{(3)} = 6 \quad g_1^4 = 4$$
$$g_2^{(1)} = 0 \quad g_2^{(2)} = 6 \quad g_2^{(3)} = 4 \quad g_2^{(4)} = 2.$$

In this case there is a one-to-one match between the source coder output and the channel coder input, and the results shown in Figure 4 reflect this fact. There is considerable improvement in the decoded error probability and there is about a 5 dB improvement obtained by using the MAP decoder at a probability of error of 0.1. These results justify the contention that for best use of the JSC decoder the input alphabet size of the channel coder should be the same as the size of the output alphabet of the source coder. To this point we have been using a MAP decoder with an encoder designed to maximize performance with a Hamming metric. In the next section we propose a general channel coder design to go with the map decoder which has the added flexibility of being able to match the size of the source coder output alphabet.

4 A Modified Convolutional Encoder

Given that the preservation of the structure in the source coder output requires the channel coder input alphabet to have a one-to-one match with the generally nonbinary source coder, we propose a general nonbinary convolutional encoder (NCE) whose input alphabet has the requisite property.

Let x_n, the input to the NCE, be selected from the alphabet $A = \{0, 1, 2, ..., N-1\}$, and let y_n, the output alphabet of the NCE, be selected from the alphabet $S = \{0, 1, 2, ..., M-1\}$. Then the proposed NCEs can be described by the following mappings

Rate 1/2 NCE: $M = N^2$; $y_n = Nx_{n-1} + x_n$

Rate 1/3 NCE: $M = N^3$; $y_n = N^2x_{n-2} + Nx_{n-1} + x_n$

Rate 2/3 NCE: $M = N^3$; $y_n = N^2x_{2n-2} + Nx_{2n-1} + x_2n$

Because of lack of space we will only describe and present the results for the rate 1/2 NCE. The description and results for the other cases can be found in [24] and are similar to the results for the rate 1/2 NCE code.

The number of bits required to represent the output alphabet of the NCE codes using a fixed length code is

$$\lceil \log_2(M) \rceil = \lceil \log_2(N^2) \rceil = \lceil 2\log_2(N) \rceil$$

Therefore in terms of rate, the rate 1/2 NCE coder is equivalent to a rate 1/2 convolutional encoder. The encoder memory in bits is $2\lceil \log_2(N) \rceil$ as each output value depends on two input values.

As an example, consider the situation when $N = 4$. Then $A = \{0,1,2,3\}$ and $S = \{0,1,2,...,15\}$. Given the input sequence x_n : 0 1 3 0 2 1 1 0 3 3 and assuming the encoder is initialized with zeros, the output sequence will be y_n : 0 1 7 12 2 9 5 4 3 15.

The encoder memory is four bits. Notice that while the encoder output alphabet is of size N^2, at any given instant the encoder can only emit one of N different symbols as should be the case for a rate 1/2 convolutional encoder. For example if $y_{n-1} = 0$, then y_n will take on a value from $\{0,1,2,...,(N-1)\}$. In general, given a value for y_{n-1}, y_n will take on a value from $\{\alpha N, \alpha N+1, \alpha N+2, ..., \alpha N+N-1\}$, where $\alpha = y_{n-1}(mod N)$. This structure can be used by the decoder to provide error protection. The encoder is shown in Figure 5.

4.1 Binary Encoding of the NCE Output

We will make use of the residual structure in the source coder output (which is preserved in the NCE output) at the receiver. However, we can also make use of this structure in selecting binary codes for the NCE output. An intelligent assignment of binary codes can improve the error correcting performance of the system.

When each allowable sequence is equally likely, there is little reason to prefer one particular

assignment over others. However, when certain sequences are more likely to occur than others, it would be useful to make assignments which increase the 'distance' between likely sequences. While, for small alphabets it is a simple matter to assign the optimum binary codewords by inspection, this becomes computationally impossible for larger alphabets. We use a rather simple heuristic which, while not optimal, provides good results.

Our strategy is to try to maximize the Hamming distance between codewords that are likely to be mistaken for one another. First we obtain a partition of the alphabet based on the fact that given a particular value for y_{n-1}, y_n can only take on values from a subset of the full alphabet. To see this, consider the rate 1/2 NCE; then the alphabet S can be partitioned into the following sub-alphabets:

$$S_0 = (0,1,2,3...,N-1)$$

$$S_1 = (N, N+1, ...,2N-1)$$

$$\vdots$$

$$S_{N-1} = \left(N(N-1), N(N-1)+1, ..., N^2-1\right)$$

where the encoder will select letters from alphabet S_j at time n if $j = y_{n-1}(\mathrm{mod}N)$. Now for each sub-alphabet we have to pick N codewords out of M $(= N^2)$ possible choices. We first pick the sub-alphabet containing the most likely letter. The letters in the sub-alphabet are ordered according to their probability of occurrence. We assign a codeword a from the list of available codewords to the most probable symbol. Then, assign the complement of a to the next symbol on the list. Therefore the distance between the two most likely symbols in the list is $K = \lceil \log_2 M \rceil$ bits. We then pick a codeword b from the list which is at a Hamming distance of $K/2$ from a and assign it and its complement to the next two elements on the list. This process is continued with the selection of letters that are$K/2^k$ away from a at the k^{th} step until all letters in the subalphabet have a codeword assigned to them.

4.2 Simulation Results

The proposed approach was simulated using the same setup as was used in the preceding simulations. We show the results for the rate 1/2 NCE coder in Figure 6 and comparisons in Figure 7. Note the dramatic improvement in performance with the rate 1/2 NCE code. At a probability of error of 0.1 the RSNR drops by less than 1 dB. For the same channel conditions the use of the (2,1,3) code results in a drop of more than 10 dB. Looking at the decoded error probabilities, even when the channel error probability is 0.25, the decoded error probability is less than 10^{-2}. This improvement has been obtained with only a minimal increase in complexity. Similar results have also been obtained for rate 1/3 and 2/3 NCE codes.

5 Conclusion

If the source and channel coder are designed in a "joint" manner, that is the design of each takes into account the overall conditions (source as well as channel statistics), we can obtain excellent performance over a wide range of channel conditions. In this paper we have presented one such design. The resulting performance improvement seems to validate this approach, with a "flattening out" of the performance curves. This flattening out of the performance curves makes the approach useful for a large variety of channel error conditions.

References

[1] C. E. Shannon, *Bell Syst. Tech. J.*, vol. 27, pp. 379–423, 623–656, 1948.

[2] J. G. Dunham and R. M. Gray, *IEEE Trans. Inform. Theory*, vol. IT-27, pp. 516–519, July 1981.

[3] E. Ayanoğlu and R. M. Gray, *IEEE Trans. Inform. Theory*, vol. IT-33, pp. 855–865, Nov. 1987.

[4] J. G. Dunham and R. M. Gray, *IEEE Trans. Inform. Theory*, vol. IT-27, pp. 516–519, July 1981.

[5] J. L. Massey, in *Communication Systems and Random Process Theory*, J. K. Skwirzynski, ed., Sijthoff and Nordhoff: Netherlands, pp. 279–293, 1978.

[6] T. C. Ancheta, Jr. Ph.D. dissertation, Dept. of Electrical Engr., Univ. of Notre Dame. Aug. 1977.

[7] K–Y Chang and R. W. Donaldson, *IEEE Trans. Commun.*, vol. COM-20, pp. 338–350, June 1972.

[8] A. J. Kurtenbach and P. A. Wintz, *IEEE Trans. Commun. Technol.*, vol. COM-17, pp. 291–302, April 1969.

[9] N. Farvardin and V. Vaishampayan, *IEEE Trans. Inform. Theory*, vol. IT-33, pp. 827–838, November 1987.

[10] D. J. Goodman and C. E. Sundberg, *Bell Syst. Tech. J.*, vol. 62, pp. 2017–2036, Sept. 1983.

[11] D. J. Goodman and C. E. Sundberg, *Bell Syst. Tech. J.*, vol. 62, pp. 2735–2764, Nov. 1983.

[12] R. C. Reininger and J. D. Gibson, *IEEE Trans. Commun.*, vol. COM-31, pp. 572–577, April 1983.

[13] R. Steele, D. J. Goodman, and C. A. McGonegal, *IEEE Trans. Commun.*, vol. COM-27, pp. 252–255, January 1979.

[14] R. Steele, D. J. Goodman, and C. A. McGonegal, *Elec. Lett.*, vol. 13, pp. 351–353, June 1977.

[15] K. N. Ngan and R. Steele, *IEEE Trans. Commun.*, vol. COM-30, pp. 257–269, January 1982.

[16] K. Sayood and J. C. Borkenhagen, *Proceedings IEEE International Conference on Communications*, June 1986, pp. 1888–1892.

[17] K. Sayood and J. C. Borkenhagen, *IEEE Transactions on Communications*, vol. COM-39, June 1991.

[18] K. Sayood and J. D. Gibson, *Proc. 22nd Annual Conference on Information Sciences and Systems*, Princeton, NJ, Mar. 1988, pp. 380-385.

[19] J. W. Modestino, D. G. Daut, and A. L. Vickers, *IEEE Trans. Commun.*, vol. COM-29, pp. 1262–1274, Sept. 1981.

[20] D. Comstock and J. D. Gibson, *IEEE Trans. Commun.*, vol. COM-32, pp. 856–861, July 1984.

[21] C. C. Moore and J. D. Gibson, *IEEE Trans. Commun.*, vol. COM-32, August 1984.

[22] K. Sayood, J. D. Gibson, and F. Liu, *Proc. 22nd Annual Asilomar Conference on Circuits, Systems, and Computers*, Nov. 1988, pp. 102-106.

[23] S. Lin and D. J. Costello, *Error Control Coding*, Prentice Hall, 1983.

[24] K. Sayood, F. Liu and J. D. Gibson, *IEEE Trans. Commun.* To be submitted.

Table 1: **Codeword Assignments**

Symbol	Code	Symbol	Code
0	0000	8	1011
1	0011	9	0111
2	1100	10	0100
3	1111	11	1000
4	1110	12	0101
5	1101	13	1001
6	0001	14	1010
7	0010	15	0110

Figure 1. Images used in simulation

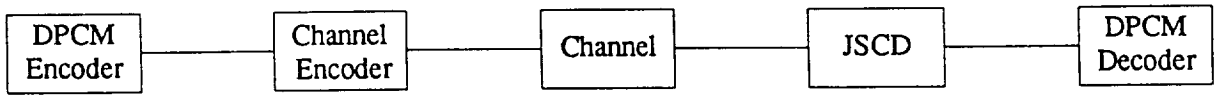

Figure 2. Block Diagram of Proposed System

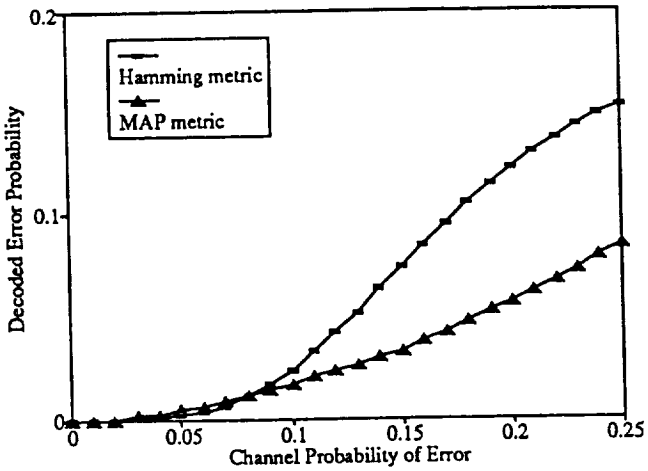

Figure 3a. Decoded error probability for the (2,1,3) convolutional coder.

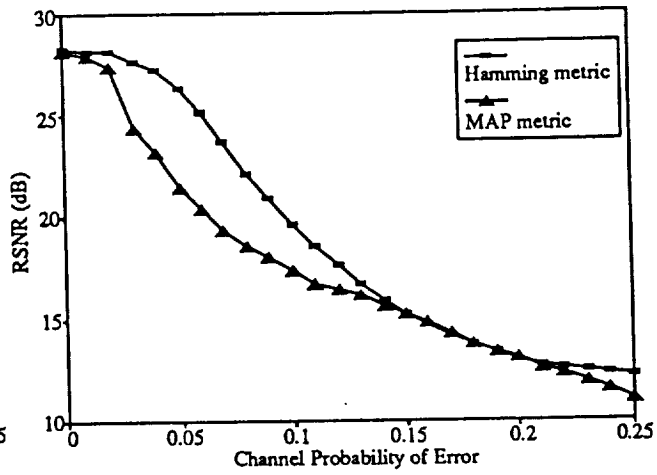

Figure 3b. RSNR for the (2,1,3) convolutional coder

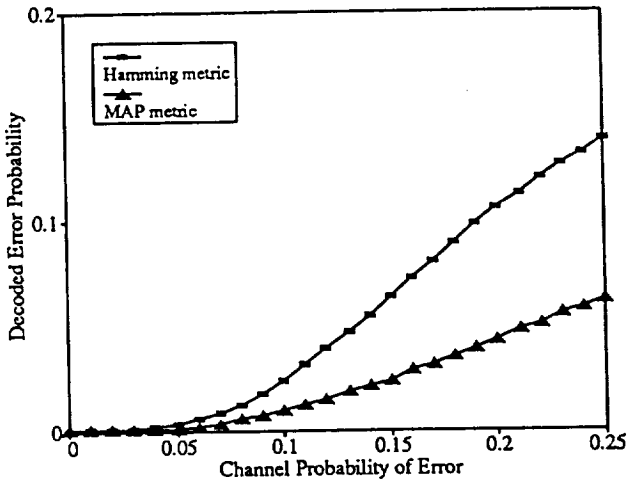

Figure 4a. Decoded error probability for the (4,2,1) convolutional coder

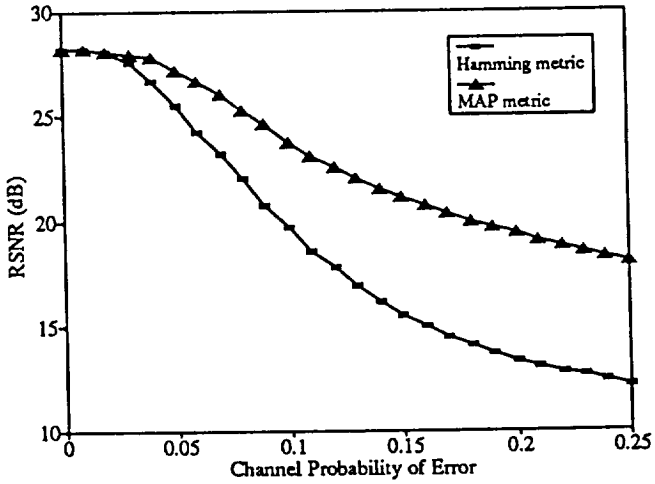

Figure 4b. RSNR for the (4,2,1) convolutional coder

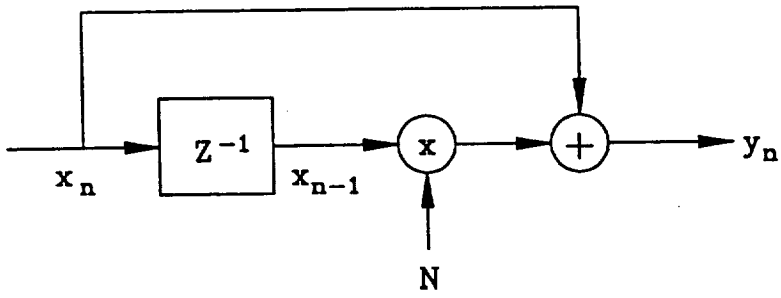

Figure 5. Rate 1/2 Nonbinary Convolutional Encoder

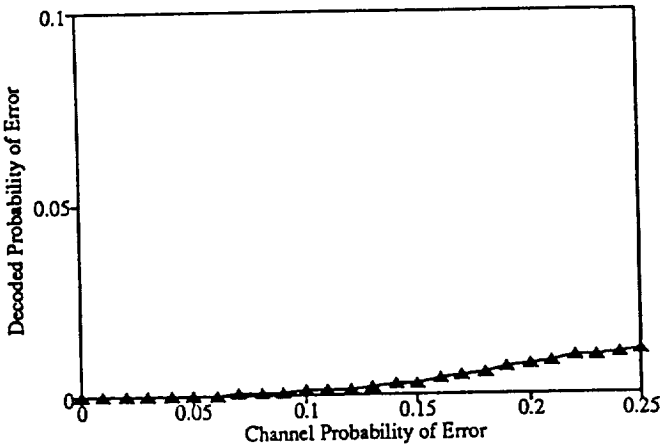

Figure 6a. Decoded vs channel error for rate 1/2 NCE

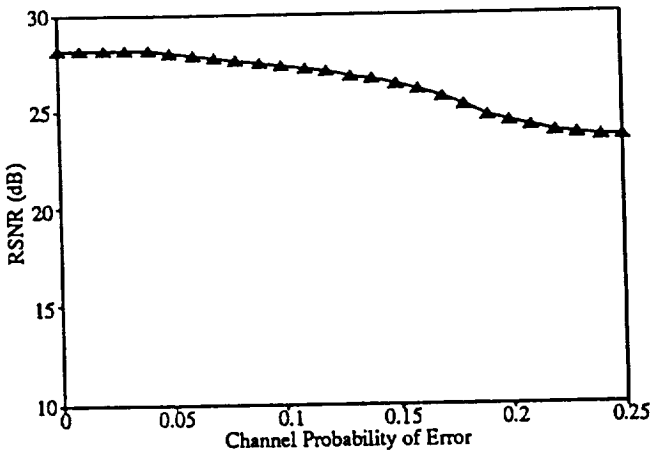

Figure 6b. RSNR for rate 1/2 NCE coder vs channel error

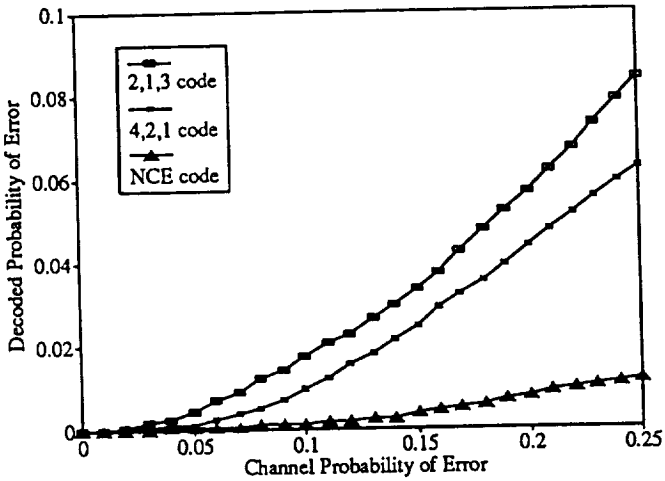

Figure 7a. Decoder error probability for the three MAP decoded systems

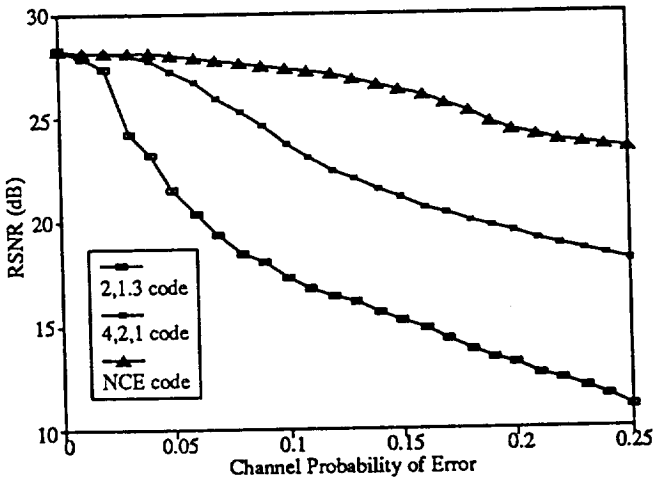

Figure 7b. RSNR vs channel error for the three MAP decoded systems