Achieving Unequal Error Protection with Convolutional Codes

D. G. Mills, D. J. Costello, Jr.,* and R. Palazzo, Jr.†
Department of Electrical Engineering
University of Notre Dame
Notre Dame, IN 46556

A Preliminary Draft of a paper to be
Submitted as a Regular Paper to the
IEEE Transactions on Information Theory

August 1994

Abstract

This paper examines the unequal error protection capabilities of convolutional codes. Both time-invariant and periodically time-varying convolutional encoders are examined. The effective free distance vector is defined and is shown to be useful in determining the unequal error protection (UEP) capabilities of convolutional codes. A modified transfer function is used to determine an upper bound on the bit error probabilities for individual input bit positions in a convolutional encoder. The bound is heavily dependent on the individual effective free distance of the input bit position. A bound relating two individual effective free distances is presented. The bound is a useful tool in determining the maximum possible disparity in individual effective free distances of encoders of specified rate and memory distribution. The unequal error protection capabilities of convolutional encoders of several rates and memory distributions are determined and discussed.

Index Terms: Unequal error protection, effective free distance vector, memory distribution, individual bit error probability.

*Dr. Mills' and Dr. Costello's work was supported by NASA Grants NAG5-557 and NAG3-1549 and NSF Grant EID90-17558

†On leave from the State University of Campinas - UNICAMP, Brazil. Dr. Palazzo's work was supported by Fundação de Amparo a Pesquisa do Estado de São Paulo - FAPESP, Brazil, under grant 92/4845-7.

1 Introduction

In conventional channel coding applications, it is assumed that the input symbols to the channel are equally likely and that the code provides essentially equal error probability to each bit. However, in some applications, certain bit position(s) in the information stream are more important than others. For example, the sign bit and high order bits of pulse coded modulation (PCM) data are more critical to system performance than the lower order bits [1]. In packet switched networks, the header information requires more error protection than the data; and in multi-user environments, some users may require more error protection than others. In mobile communications, due to severe constraints imposed by the channel, hybrid modulation is needed to guarantee reliable transmission and in addition provides unequal error protection to the data being transmitted [2]. Systems in which some information is non-essential enhancement information, e.g. embedded coding schemes and high definition television (HDTV), are also potential application environments. Encoders which provide more than one level of error protection to information bits are called unequal error protection (UEP) encoders.

Since the introduction of linear unequal error protection (LUEP) codes by Masnick and Wolf [1], many researchers have derived new results for classes of linear block codes with unequal error protection for *single positions in codewords* or *for single positions in the input information digits.* These classes of codes consist of nonsystematic cyclic UEP codes [2], codes derived from difference sets [3], iterative and concatenated designs of UEP codes [4],cyclic code classes [5], and LUEP codes derived from shorter codes [6].

In this paper, we examine the unequal error protection capabilities of convolutional codes by presenting classes of convolutional codes which satisfy the basic property of UEP codes, that is, provide unequal error protection for each *input information digit.* The new classes contain both time-invariant convolutional codes (TICC) and periodically time-varying convolutional codes (PTVCC).

The work done in [1]-[7] established the existence of systematic procedures for the construction of LUEP block codes. Those methods used either the generator matrix or the parity-check matrix for code construction with an assumed optimal decoding method, typically majority logic decoding.

In contrast with the previous LUEP block codes, the UEP convolutional encoders presented in this paper lack algebraic structure. For that reason, good encoders are found by a computer search procedure. The assumed decoding method is Viterbi decoding for short-constraint-lengths, or sequential decoding for long constraint-lengths.

The paper is organized as follows. In Section 2, convolutional codes are briefly reviewed. In Section 3, the effective free distance vector is presented as an alternative parameter to the free distance for determining the UEP capabilities on encoders. Section 4, discusses classes of convolutional codes which satisfy the UEP property. Section 5 presents a modified transfer function from which the UEP capabilities of an encoder can be calculated. A method to

Example 1 *Consider the rate $r = 2/3$ convolutional encoder with $\mathbf{M} = (1,2), K = 3$, and encoding matrices*

$$\mathbf{G}_0 = \begin{bmatrix} 1 & 0 & 1 \\ 0 & 1 & 1 \end{bmatrix}; \mathbf{G}_1 = \begin{bmatrix} 0 & 1 & 1 \\ 0 & 0 & 1 \end{bmatrix}; \mathbf{G}_2 = \begin{bmatrix} 0 & 0 & 0 \\ 1 & 0 & 1 \end{bmatrix}$$

This encoder is shown in Fig. 2. The free distance is 4.

An (n, k, m) periodically time-varying convolutional encoder with period P can be represented by the encoding equation

$$\mathbf{v}_t = \mathbf{u}_t \mathbf{G}_0^{(t)_P} \oplus \mathbf{u}_{t-1} \mathbf{G}_1^{(t)_P} \oplus \ldots \oplus \mathbf{u}_{t-m} \mathbf{G}_m^{(t)_P}, \tag{5}$$

where $(t)_P$ denotes the value of t modulo P, and the encoding matrices $\mathbf{G}_i^{(t)_P}, i = 0, 1, \ldots, m$, are $(k \times n)$ binary matrices. It is assumed that the non-zero input occurs at time 0. For example, the periodically time-varying convolutional encoder with period 2 and encoding matrices

$$\mathbf{G}_0^{(0)} = \begin{bmatrix} 1 & 1 & 1 \\ 0 & 1 & 0 \end{bmatrix}; \mathbf{G}_1^{(0)} = \begin{bmatrix} 1 & 0 & 1 \\ 1 & 1 & 0 \end{bmatrix}$$

$$\mathbf{G}_0^{(1)} = \begin{bmatrix} 0 & 1 & 1 \\ 0 & 0 & 0 \end{bmatrix}; \mathbf{G}_1^{(1)} = \begin{bmatrix} 1 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

produces the code sequence

$$\mathbf{v} = (111\ 101\ 001\ 001\ 101\ 100\ldots)$$

for the input sequence

$$\mathbf{u} = (10\ 11\ 01\ 01\ 11\ 10\ldots).$$

3 The Effective Free Distance Vector

It is convenient to define an effective free distance vector, $\mathbf{d_{eff}}$, as an alternative to the free distance as a primary performance parameter for the linear unequal error protection codes. The performance criterion based on the effective free distance vector is equivalent to the separation vector concept of (n, k) linear codes introduced by Dunning and Robbins [7]. Similar vectors have been proposed in [14], [15], [8], [9], [10], [25].

Definition 1 *For an (n, k, m), $k \neq 1$, time-invariant convolutional encoder, and an (n, k, m) periodically time-varying convolutional encoder, the effective free distance vector is defined as the k-dimensional vector*

$$\mathbf{d_{eff}}(C) = (d_1, d_2, \ldots\ldots, d_k) \tag{6}$$

The non-zero path through the trellis with weight 3 *is shown by dotted lines. The non-zero paths with weight* 4 *are shown with dark lines. All other paths have weight greater than* 4. *It can be seen that the weight* 3 *path is created by an input vector sequence that is non-zero only in the first input bit position,* (01 00). *When the input sequences are non-zero in the second position, the minimum weight of any path is* 4.

It is important to recognize that the first "1" in the j-th position does not necessarily occur at time zero. For instance, the input sequence $10, 01, 00$ is one of the sequences that must be considered when determining the second effective free distance, d_1, of an encoder with two input lines ($k = 2$), and $\mathbf{M} = (1, 1)$. The input sequence $10, 10, 00$ need not be considered when determining d_1.

It should be noted that the effective free distance vector is dependent on the encoder realization of a code. An example that demonstrates the dependence follows.

Example 3 *The encoder described in Example 2 and the encoder shown in Figure 5 with encoding matrices*

$$\mathbf{G}_0 = \begin{bmatrix} 1 & 0 & 1 \\ 1 & 1 & 1 \end{bmatrix}; \mathbf{G}_1 = \begin{bmatrix} 0 & 1 & 1 \\ 1 & 0 & 1 \end{bmatrix} \tag{9}$$

are different realizations of the same code. Figure 6 shows the trellis for the encoder in Figure 5. It can be seen that the effective free distance vector for the encoder in this example is $\mathbf{d_{eff}} = (3, 3)$, *which differs from the effective distance vector of the equivalent encoder realization discussed in Example 2.*

4 Classes of UEP Convolutional Codes

We begin this section by restricting the class of convolutional encoders that can be used as UEP codes.

Theorem 2 *An* $(n, 1, m)$ *time-invariant binary convolutional encoder can not provide unequal error protection.*

Proof. Consider an $(n, 1, m)$ convolutional encoder with its corresponding trellis diagram. Since the code is linear, without loss of any generality, assume the all zero sequence is transmitted. A decoding error occurs at time $t = k$ if there exists a path diverging from the correct path at time k or prior to it and reemerging later. Let $\mathbf{d} = \{d_0, d_1, d_2, ...\}$ be the ordered set of Hamming weights of all paths satisfying the above decoding error.

Now, assume that a decoding error occurs at time $t = k'$ with $k' \neq k$. Let $\mathbf{d}' = \{ d'_0, d'_1, d'_2, ...\}$ be the ordered set of Hamming weights of all paths that diverged at or prior to k' and reemerged later.

Since the underlying Markov process is stationary, the statistics associated with the error event at time $t = k$ and $t = k'$ are the same. Therefore, the sets $\mathbf{d}$ and $\mathbf{d}'$ are equal and any information digit has the same free distance.□

will be described and used to develop an upper bound on the average bit error probability for a specific input bit position. A method to calculate the modified transfer function for time-varying and time-invariant encoders is described. Finally, several examples are presented.

5.1 A modified transfer function

The standard two-variable transfer function [11] has the form

$$T(X,Y) = \sum_{d=d_{free}}^{\infty} \sum_{b=1}^{\infty} A_{b,d}.X^d.Y^b. \tag{10}$$

The average bit error probability for a specific transfer function is bounded by

$$P_b(E) < \left(\frac{1}{k}\right).\sum_d B_d.P_d, \tag{11}$$

where $B_d = \sum_b b.A_{b,d}$ is the total number of non-zero information bits associated with all codewords of weight d, and $P_d = \left(\sqrt{4p(1-p)}\right)^d = D^d$, where D is the Bhattacharyya function [22]. For the sake of simplicity, we assume a binary symmetric channel with crossover probability p.

When the individual bit error probability is desired for each of the k input positions, then the split-state diagram must be modified before Mason's formula is applied. Each branch label has the new form

$$X^i Y_0^{j_0} Y_1^{j_1} \cdot\cdot\cdot\cdot\cdot Y_{k-1}^{j_{k-1}}, \tag{12}$$

where j_l is equal to the input bit in the $l\text{-}th$ position, and i is the Hamming weight of the branch output. Obviously, the sum of the j_l's is the Hamming weight of the input vector. The transfer function is then calculated. The resulting modified transfer function is

$$T(X,Y_0,Y_1,....,Y_{k-1}) = \sum_{d=d_{free}}^{\infty} \sum_{j=1}^{j_d} C_{d,j} X^d Y_0^{b_{0,j}} Y_1^{b_{1,j}} \cdot\cdot\cdot\cdot\cdot Y_{k-1}^{b_{k-1,j}}, \tag{13}$$

where $C_{d,j}$ is the number of paths associated with the $j\text{-}th$ input sequence distribution of 1's that generates code vectors of weight d, j_d is the number of distinct input sequence distributions that generate code vectors of weight d, and the entity $b_{0,j}, b_{1,j}, ..., b_{k-1,j}$ represents a particular input sequence distribution of 1's. The bound for the individual bit error probabilities is then

$$P_b^{(i)}(E) < \sum_d B_d^{(i)}.P_d, \text{ for } 0 \le i \le k-1, \tag{14}$$

where $P_b^{(i)}(E)$ is the probability that a bit located in the $i\text{-}th$ position of the input vector is decoded incorrectly, and $B_d^{(i)} = \sum_{j=1}^{j_d} b_{i,j}.C_{d,j}$ is the total number of 1's in bit position i contained in all input vectors that generate code vectors of weight d. Note that the

It is assumed that the discrete-time evolution of these systems follows the following pattern: system L_1 (convolutional encoder) is **on** in the time interval $i \leq k < i+1$ and system L_2 (convolutional encoder) is **on** in the time interval $i+1 \leq k < i+2$, for any i, i an integer. It can be shown that the combination of these two linear systems (convolutional encoders) results in a linear system (convolutional encoder).

The state and output equations of the resulting linear system can be described mathematically with the previous state and output equations, by the following reasoning: the states ε_1 (ε_2) belonging to system L_1 (L_2), at time $t = k+1$ can be reached from the states ε_2 (ε_1) belonging to system L_2 (L_1) at time $t = k$, by applying the transition matrix A_1 (A_2), corresponding to system L_1 (L_2), or from the initial condition matrix B_1 (B_2); the output equation T_1 (T_2) of system L_1 (L_2) is obtained by multiplying the output condition matrix C_1 (C_2) by the state matrix ε_2 (ε_1) of system L_2 (L_1).

Mathematically,

$$\text{System } L_1 \Longleftrightarrow \begin{cases} \varepsilon_1(k+1) = A_1(k).\varepsilon_2(k) + B_1(k) \\ \\ T_1(k) = C_1(k).\varepsilon_2(k) \end{cases} \tag{18}$$

$$\text{System } L_2 \Longleftrightarrow \begin{cases} \varepsilon_2(k+1) = A_2(k).\varepsilon_1(k) + B_2(k) \\ \\ T_2(k) = C_2(k).\varepsilon_1(k) \end{cases} \tag{19}$$

The total output equation is

$$T(k) = T_1(k) + T_2(k) \tag{20}$$

Solving the equations for $\varepsilon_1(k)$ and $\varepsilon_2(k)$, we have

$$\begin{aligned} \varepsilon_1 &= (I - A_1A_2)^{-1}(A_1B_2 + B_1) \\ \\ \varepsilon_2 &= A_2(I - A_1A_2)^{-1}(A_1B_2 + B_1) + B_2 \end{aligned} \tag{21}$$

The modified transfer function of System 1 is

$$T_1(X, Y_0, \ldots, Y_{k-1}) = C_1A_2(I - A_1A_2)^{-1}(A_1B_2 + B_1) + B_2, \tag{22}$$

the modified transfer function of System 2 is

$$T_2(X, Y_0, \ldots, Y_{k-1}) = C_2(I - A_1A_2)^{-1}(A_1B_2 + B_1), \tag{23}$$

and the overall modified transfer function is

$$T(X, Y_0, \ldots, Y_{k-1}) = C_1A_2(I - A_1A_2)^{-1}(A_1B_2 + B_1) + C_1B_2 + C_2(I - A_1A_2)^{-1}(A_1B_2 + B_1) \tag{24}$$

The system matrices for each convolutional encoder are easily determined from the split-state diagram. Let $a_1(i,j)$ be the element of A_1 in row i and column j, $b_1(i)$ be the

$$A_2 = \begin{bmatrix} X^2Y_1 & XY_1 & X^3Y_1 \\ X^2Y_0 & XY_0 & XY_0 \\ X^2Y_0Y_1 & XY_0Y_1 & XY_0Y_1 \end{bmatrix} ; B_2 = \begin{bmatrix} X^2Y_1 \\ Y_0 \\ X^2Y_0Y_1 \end{bmatrix} ; C_2 = \begin{bmatrix} X^2 & X & X^3 \end{bmatrix}$$

Using these matrices in (24), we determine that the transfer function is

$$T(X, Y_0, Y_1) = 2X^2Y_0+X^2Y_0^2+X^3Y_0Y_1+X^4Y_1+2X^4Y_0Y_1+X^4Y_0^2+X^4Y_0^3+2X^4Y_0^2Y_1+X^4Y_0^3Y_1+\ldots. \quad (26)$$

Then, the average probability of a bit error in position 0 is

$$P_b^{(0)} \leq 4P_2 + P_3 + 14P_4 + \ldots.$$

Similarly, the average bit error for the second input position is

$$P_b^{(1)} \leq P_3 + 6P_4 + \ldots.$$

Therefore, $\mathbf{d_{eff}} = (2, 3)$. Finally, the overall average bit error rate is

$$P_b(E) = \frac{1}{2}P_b^{(0)} + \frac{1}{2}P_b^{(1)} \leq 2P_2 + P_3 + 10P_4 + \ldots$$

Example 5 *Now consider the time-invariant case by assuming that system L_1 is the only system.*

The previous results may be used to determine the transfer function by setting $A_2 = A_1$, $B_2 = B_1$, and $C_2 = C_1$ in (24). The total transfer function T is then

$$T = 2C_1(I - A_1)^{-1}B_1 \quad (27)$$

Evaluating (27), we have

$$\begin{aligned} T = \; & X^3Y_0+ \\ & X^4(2Y_0Y_1 + Y_0^2 + Y_0^2Y_1^2)+ \\ & X^5(Y_0 + 3Y_0Y_1^3 + 3Y_0^2Y_1 + Y_0^3 + 3Y_0^3Y_1^2 + 2Y_0^2Y_1^3 + Y_0^4Y_1^3)+ \end{aligned}$$

Then, the bounds on the individual bit error probabilities are

$$P_b^{(0)} \leq P_3 + 6P_4 \ldots,$$

and

$$P_b^{(1)} \leq 4P_4 + \ldots.$$

Therefore, $\mathbf{d_{eff}} = (3, 4)$. We remind the reader that the encoder in this example is the same encoder discussed in Example 2. The transfer function indicates that there is one code sequence of Hamming weight 3 which is generated by an input sequence with one 1 in the

6 Bounds

In this section, a bound on the individual effective free distances for time-invariant convolutional encoders is derived. Evaluating the bound is a useful tool in identifying encoder configurations which possess potential UEP capabilities. In addition, it allows a comparison between the effective free distance of a specific encoder and the theoretically optimal effective free distance. First, a bound on the Hamming weight of the sum of two vectors with known Hamming weights is presented. Then this bound is applied to effective free distances and the implications are discussed.

Let $\mathbf{x}$ and $\mathbf{y}$ be n-bit binary vectors and let $\mathbf{z}$ be the modulo 2 sum of $\mathbf{x}$ and $\mathbf{y}$,

$$\begin{aligned} \mathbf{z} &= (z_1, z_2, ..., z_n) \\ &= x \oplus y \\ &= (x_1 \oplus y_1, x_2 \oplus y_2, ..., x_n \oplus y_n) \end{aligned} \tag{28}$$

Assume that the Hamming weights of $\mathbf{x}$ and $\mathbf{y}$ are known and are w_x and w_y, respectively. It can be shown that the Hamming weight of $\mathbf{z}$ is upper bounded by the following relationship

$$w_z \leq \min\{n - w_x, w_y\} + \min\{n - w_y, w_x\}. \tag{29}$$

The proof of the bound in (29) is given below.

There are two cases which result in $z_i = 1$ and which contribute to the Hamming weight of $\mathbf{z}$. *Case 1* occurs when x_i is 1 and y_i is 0; *Case 2* occurs when x_i is 0 and y_i is 1. The Hamming weight of $\mathbf{z}$ is equal to the total number of bit positions in which either of the two cases appears. Therefore, w_z can be upper bounded by the sum of the maximum number of occurrences of *Case 1* and the maximum number of occurrences of *Case 2*. The number of bit positions in which *Case 1* occurs can be no greater than the minimum of the number of 1 's in $\mathbf{x}$ and the number of 0's in $\mathbf{y}$. Similarly, the number of bit positions in which *Case 2* occurs can be no greater than the minimum of the number of 0's in $\mathbf{x}$ and the number of 1's in $\mathbf{y}$. Therefore,

$$w_z \leq \min\{n - w_x, w_y\} + \min\{n - w_y, w_x\}. \tag{30}$$

The bound in (29) can be applied to a convolutional encoder and provides the basis for a bound on the effective free distance of a particular input line as a function of the effective free distance of another line.

Recall that a rate $R = k/n$ convolutional encoder with input (message) sequence $\mathbf{u} = (\mathbf{u}_0, \mathbf{u}_1, ...)$, code sequence $\mathbf{v} = (\mathbf{v}_0, \mathbf{v}_1, ...)$, and memory distribution $\mathbf{M} = (m_0, m_1, ..., m_{k-1})$, with $m_0 \leq m_1 \leq ..... \leq m_{k-1}$, can be represented by the equation

$$\mathbf{v}_j = \mathbf{u}_j.\mathbf{G}_0 \oplus \mathbf{u}_{j-1}.\mathbf{G}_1 \oplus ..... \oplus \mathbf{u}_{j-m_{k-1}}.\mathbf{G}_{m_{k-1}}, \tag{31}$$

where $\mathbf{u}_i$ is a binary *k-tuple*, $\mathbf{v}_i$ is a binary *n-tuple*, and $\mathbf{G}_i$ is a $(k \times n)$ binary matrix. Recall that the concatenated encoding matrix, $\mathbf{G}$, is the concatenation of the matrices $\mathbf{G}_i$, $0 \leq i \leq m_{k-1}$, i.e., $\mathbf{G} = [\mathbf{G}_0|\mathbf{G}_1|\cdots|\mathbf{G}_{m_{k-1}}]$.

line j. The original encoder has the encoder matrix

$$\mathbf{G} = [\mathbf{G}_0|\mathbf{G}_1|\cdots|\mathbf{G}_{m_{k-1}}] \tag{38}$$

$$= \begin{bmatrix} \mathbf{r}_0 \\ \mathbf{r}_1 \\ \vdots \\ \mathbf{r}_{k-1} \end{bmatrix} \tag{39}$$

Using the appropriate periodic input sequences, we can form two vectors

$$\mathbf{r'}_i = \underbrace{[\mathbf{r}_i|\mathbf{r}_i|\ldots|\mathbf{r}_i]}_{b \text{ times}} \tag{40}$$

and

$$\mathbf{r'}_j = \underbrace{[\mathbf{r}_j|\mathbf{r}_j|\ldots|\mathbf{r}_j]}_{c \text{ times}}, \tag{41}$$

which are valid code sequences.

Note that $w'_i = w_H(\mathbf{r'}_i) = bw_i$ and $w'_j = w_H(\mathbf{r'}_j) = cw_j$. From the definition of the effective free distance,

$$d_j \leq w_H(\mathbf{r'}_i \oplus \mathbf{r'}_j). \tag{42}$$

Let $N = n \max[b(m_i+1), c(m_j+1)]$. From the vector bound in (29),

$$w_H(\mathbf{r'}_i \oplus \mathbf{r'}_j) \leq \min\left\{N - w'_i, w'_j\right\} \tag{43}$$

$$+ \min\left\{N - w'_j, w'_i\right\}. \tag{44}$$

So,

$$d_j \leq \min\left\{N - w'_i, w'_j\right\} \tag{45}$$

$$+ \min\left\{N - w'_j, w'_i\right\} \tag{46}$$

$$\leq [N - w'_i] + [N - w'_j] \tag{47}$$

$$= 2N - w'_i - w'_j. \tag{48}$$

Rewriting,

$$d_j + w'_j + w'_i \leq 2N \tag{49}$$

or

$$d_j + bw_i + cw_j \leq 2N. \tag{50}$$

Since $d_j \leq w_j$ and $d_i \leq w_i$, the bound can be loosened to

$$bd_i + (c+1)d_j \leq 2N = 2n \max[b(m_i+1), c(m_j+1)] \tag{51}$$

in the first d_0 positions.

Situation a) implies than $s_j^0 \leq d_0 - \lceil \frac{d_0}{2} \rceil$, to ensure that d_0 is the effective distance of row 0. In addition, y_j^0 is obviously upperbounded by $n(m_j + 1) - d_0$, By definition, $d_j \leq w_H(rowj \oplus row0)$ and $w_H(rowj \oplus row0) = s_j + y_j$, so

$$d_j \leq s_j^0 + y_j^0. \tag{54}$$

Using the upper bounds for s_j^0 and y_j^0, we obtain

$$d_j \leq (d_0 - \lceil \frac{d_0}{2} \rceil) + n(m_j + 1) - d_0. \tag{55}$$

Simplifying,

$$d_j + \lceil \frac{d_0}{2} \rceil \leq n(m_j + 1). \tag{56}$$

Allowing $j = 1$ and defining $X_1 = \lceil \frac{d_0}{2} \rceil$, we have

$$d_1 + X_1 \leq n(m_1 + 1) \tag{57}$$

for $j = 1, \ldots, k-1$.

Because l_j^0 is the number of 1's in the first d_0 positions, and situation b) assumes that there are at least $\lceil \frac{d_0}{2} \rceil$ 0's in those positions, Situation b) implies that $d_0 - \lceil \frac{d_0}{2} \rceil \geq l_j \geq 0$. Also, because the Hamming weight of row j is no less than d_j,

$$d_j \leq l_j + y_j \tag{58}$$

It then follows that

$$d_j \leq d_0 - \lceil \frac{d_0}{2} \rceil + n(m_j + 1) - d_0, \tag{59}$$

which simplifies to

$$d_j + \lceil \frac{d_0}{2} \rceil \leq n(m_j + 1). \tag{60}$$

We are now left with a residual block code of rate $\frac{k-1}{n(m_{k-1}+1)-d_0}$ with a row weight vector, $\mathbf{w} \geq (w_1 - (d_0 - X_1), \ldots, w_j - (d_0 - X_1), \ldots, w_{k-1} - (d_0 - X_1))$, and an effective distance vector $\mathbf{d} = (d_1^*, \ldots, d_j^*, \ldots, d_{k-1}^*)$. Repeating the procedure, we rearrange the columns so that the first row has only ones in the first $w_1 - (w_0 - X_1)$ positions, and only zeros in the remaining $n(m_{k-1}) - d_0 - (w_1 - (d_0 - X_1)) = n(m_{k-1}) - w_1 - X_1$ positions. Then for $j = 2, \ldots, k-1$, the portion of the original row j that belongs to the residual code can have either: a) $\geq \lceil \frac{w_1 - (d_0 - X_1)}{2} \rceil$ 1's or b) $\geq \lceil \frac{w_1 - (d_0 - X_1)}{2} \rceil$ 0's in the first $w_1 - (d_0 - X_1)$ positions.

Again, more variables are defined. Let y_j^1 be the number of 1's in the last $n(m_j + 1) - d_0 - (w_1 - (d_0 - X_1))$ positions of row j, s_j^1 be the number of 1's in the first $w_1 - (d_0 - X_1)$ positions of row $j \oplus$ row 1, and l_j be the number of 1's in the first $w_1 - (d_0 - X_1)$ positions of row j.

When the non-zero input to a convolutional encoder has length h, the encoder can be considered as a block code with 2^{hk} codewords of length $h(m_{k-1}+1)$. Therefore, the free distance of the code may be upperbounded [29] by

$$d_{free} \leq \frac{n(m_{k-1}+h)}{2}\frac{2^{hk}}{2^{hk}-1}, h = 1, 2, \cdots. \tag{67}$$

The effective distance for a specific input line, j, is the minimum Hamming weight among codewords that are associated with inputs that contain at least one 1 on that input line. A code generated by the inputs of length h which have at least one 1 on line j is called the restricted block code, $\mathcal{C}_j^h$. The set of such inputs and outputs may be considered as a series of block codes, similar to the approach used for the Plotkin bound. We define C_j^h as the number of codewords in the restricted block code $\mathcal{C}_j^h$. Then, C_j^h is equal to the total number of codewords in the unrestricted block code with the same size input vectors minus the number of codewords that are all-zero on line j, or

$$C_j^h = 2^{hk} - 2^{h(k-1)} \tag{68}$$

or

$$C_j^h = 2^{h(k-1)}(2^h - 1). \tag{69}$$

Then, using the bound in (67), d_j is upper-bounded by

$$d_j \leq \frac{n(m_{k-1}+h)}{2}\frac{2^{h(k-1)}(2^h-1)}{2^{h(k-1)}(2^h-1)-1}, h = 1, 2, ... \tag{70}$$

for $j = 0, \cdots, k-1$. Note that each effective distance is subject to the same bounding value, i.e., the bound is independent of j.

9 Results

A non-exhaustive search for encoders that provide unequal error protection was conducted. The notation used for the encoding matrices is as follows. For each $\mathbf{G}_i$, the rows are given in octal representation, aand are separated by commas. For instance, an entry of 370,037 in the column labeled $\mathbf{G}_0$ means that the first row of G_0 is 11 111 000 and the second row is 00 011 111. The notation y^l indicates that octal digit y appears in l consecutive places in the row. For example, the entry $5^2 7^3$ represents a row equal to 55777 = 101 101 111 111 111. Tables 1 and 2 give the result for rate 2/3 and rate 2/4 encoders, respectively. A primary goal of the searches was to find encoders with at least one effective distance greater than the free distance of the optimal encoder of the same rate and memory order. A decrease in free distance is acceptable. In Table 1, there are four instances in which the higher effective free distance is larger than the optimal free distance for rate 2/3 encoders with the same state complexity. For $\mathbf{M} = (1,1)$, $\mathbf{d_{eff}} = (3,4)$, and $d_{free} = 3$ for the time-invariant convolutional encoder, and $d_{free} = 4$ for the time-varying convolutional encoder as shown in

5, 6,7, and 8 are good UEP encoders in the sense that they have the best (lexicographically) effective free distance vector among the encoders with the same rate and state complexity. These encoders were found by a heuristic search procedure with its main goal being that of finding only the best effective free distance vector. An algorithm [16] was later developed to find similar codes. We show in Table 9 two examples of the disparity between the components of the effective free distance vector of PTVCC with period 2. The first example shows encoders with rate $r = 1/2$, and memory $m = 4$, and the second example shows encoders with rate $r = 1/2$, and $m = 7$.

Bit error rate simulations were performed to verify that the effective free distance is an appropriate measure of the UEP capabilities of an encoder. Figures 8 and 9 show the bit error rate (BER) plots for the $R = 2/3$, $\mathbf{M} = (2,2)$ encoder with $\mathbf{d_{eff}} = (4,6)$ and the $R = 2/4$, $\mathbf{M} = (1,2)$ encoder with $\mathbf{d} = (6,7)$, respectively, using Viterbi decoding with soft decision decoding. Three sets of data points are shown in each plot. The data points described by the 'x' are the (simulated) bit error rate for input line 0. Similarly, the data points described by the 'o' are the (simulated) bit error rate for input line 1. The overall BERs are marked by '*'s. For cases in which BER was lower than 10^{-7}, data points do not appear. It is seen that the lower BER for a specific signal to noise ratio (SNR) is achieved by the input position with the larger effective free distance. That is, the effective free distance is a valid indication of the UEP capabilities of the encoders.

10 Conclusions

This paper discussed the unequal error protection capabilities of time-invariant and time-varying convolutional encoders. The effective free distance vector was defined as the performance parameter of interest, and a method to calculate the modified transfer function, and therefore the effective free distance vector, of an encoder was presented. An upper bound on the effective free distance vector for time-invariant encoders was derived. The bound is not tight in all cases, but it provides a quick method to evaluate if unequal error protection is possible for specific encoder configurations before searches are performed. Results of computer searches for encoders which provide unequal error protection were listed. Typically, an increase in the effective free distance of one position is accompanied by a reduction in the effective free distance of another position, relative to the free distance of the optimal encoder. However, a number of encoders that maintain the optimal free distance while providing unequal error protection were found.

[14] R. Palazzo, Jr., "On the linear unequal error protection convolutional codes," *IEEE Global Telecommunications Conference,* Dec. 1986.

[15] R. Palazzo, Jr., "Linear unequal error protection convolutional codes," *IEEE Intl. Symp. on Inform. Theory,* Brighton, England, 1985.

[16] R. Palazzo, Jr., and R. C. F. Cruz, "A new search algorithm for good unequal error protection convolutional codes," Proceedings of the *Intl. Symp. on Signals, Systems, and Electronics - URSI,* Germany, 1989.

[17] R. Palazzo, Jr., "An upper bound on the average distortion measure of cyclostationary sequences via dynamic programming," *Proceedings of the Intl. Symp. on Operational Research, Porto Alegre, Brazil, 1986.*

[18] R. Palazzo, Jr., and K. V. O. Fonseca, "Periodically time-varying trellis coded modulation," *Proceedings of the Intl. Symp. on Inform. and Coding Theory,* Campinas, Brazil, 1987.

[19] R. Palazzo, Jr., and K. V. O. Fonseca, "Unequal error protection of superlinear time-varying trellis coded modulation," *Proceedings of the 4-th Joint Sweden-USSR Intl Workshop on Inform. Theory, Visby, Sweden, 1989.*

[20] R. Palazzo, Jr., "A time-varying convolutional encoder better than the best time-invariant encoder," *IEEE Trans. Inform. Theory,* vol.IT-30, pp. 1109-1110, May 1993.

[21] R. Palazzo, Jr., "A network flow approach to convolutional codes," *IEEE Trans. Commun.,* submitted for publication Sept. 1993.

[22] A. J. Viterbi, and J. K. Omura, *Principles of Digital Communications and Coding,* MacGraw-Hill, 1979.

[23] J. K. Omura, and M. K. Simon, "Generalized transfer function techniques," *Jet Propulsion Laboratories Technical Report,* 1981.

[24] A. Shiozaki, "Unequal error protection of PCM signals by self-orthogonal convolutional codes," *IEEE Trans. on Commun.,* vol.COM-, pp. , .

[25] P. Piret, *Convolutional Codes: An Algebraic Approach,* Cambridge, Mass. MIT-Press, 1988.

[26] J. M. Wozencraft, and I. M. Jacobs, *Principles of Communication Engineering,* John Wiley & Sons, 1965.

[27] M. E. Hellman, and J. Raviv, "Probability of error, equivocation, and the Chernoff bound," *IEEE Trans. Inform. Theory,* vol.IT-16, pp. 368-372, July 1970.

11 Tables

Rate r=2/3									
M. Alloc.	**Bound**		**Optimal**	**Effect. Free Dist.**	**Encoding Matrices**				
M	d_0	d_1	d_{free}	$\tilde{d}_{eff}$	G_0	G_1	G_2	G_3	G_4
$(1,1)$	3	4	3	$(3,4)$	$3,6$	$2,5$			
$(1,2)$	4	6	4	$(4,5)$	$5,3$	$3,1$	$0,5$		
	3	6	4	$(3,5)$	$2,5$	$3,7$	$0,5$		
$(1,3)$	5	7	5	$(5,5)$	$3,4$	$7,6$	$0,7$	$0,5$	
	4	8	5	$(4,6)$	$6,3$	$3,7$	$0,5$	$0,4$	
$(1,4)$	5	9	6	$(5,6)$	$7,5$	$5,1$	$0,3$	$0,3$	$0,6$
	4	9	6	$(4,6)$	$3,5$	$5,1$	$0,7$	$0,2$	$0,6$
$(2,2)$	5	6	5	$(5,5)$	$5,3$	$4,5$	$6,3$		
	4	6	5	$(4,6)$	$1,7$	$5,2$	$1,6$		
$(2,3)$	6	7	6	$(6,6)$	$5,3$	$7,1$	$3,5$	$0,5$	
	5	8	6	$(5,6)$	$1,3$	$5,2$	$6,1$	$0,3$	
	4	8	6	$(4,6)$	$4,0$	$2,3$	$5,1$	$0,7$	

Table 1: Time-Invariant UEP Encoders

Unit Memory Encoders				
Rate	**Encoding Matrices**		**Eff. Free Dist.**	**Free Dist.**
k/n	G_0	G_1	$\tilde{d}_{eff}$	d_{free}
2/5	31, 26	25, 33	(6, 7)	6
2/8	370, 037	174, 237	(10, 11)	10
2/11	3163, 1755	1077, 3760	(14, 15)	14
2/14	37700, 03477	37760, 01777	(18, 19)	18
2/17	003777, 377700	201777, 177760	(22, 23)	22
2/20	3777600, 0017777	0777740, 3007777	(26, 27)	26

Table 3: Time-Invariant UEP Encoders

Memory Elements m = 2				
Period 2				
Rate	**Encoding Matrices**		**Eff. Free Dist.**	**Free Dist.**
k/n	*Encoder*1	*Encoder*2	$\tilde{\mathbf{d}}_{\mathbf{eff}}$	d_{free}
1/4	$5^2 7^2$	57^3	(10, 11)	10
1/7	$5^3 7^4$	$5^2 7^5$	(18, 19)	18
1/10	$5^4 7^6$	$5^3 7^7$	(26, 27)	26
1/13	$5^5 7^8$	$5^4 7^9$	(34, 35)	34
1/16	$5^6 7^{10}$	$5^5 7^{11}$	(42, 43)	42
1/19	$5^7 7^{12}$	$5^6 7^{13}$	(50, 51)	50
1/22	$5^8 7^{14}$	$5^7 7^{15}$	(58, 59)	58

Table 5: Time-Varying UEP Encoders

Memory Elements m = 3				
Period 2				
Rate	Encoding Matrices		Eff. Free Dist.	Free Dist.
k/n	$Encoder1$	$Encoder2$	$\tilde{d}_{eff}$	d_{free}
1/8	$11^2, 13^2, 17^4$	$13^3, 15^3, 17^2$	(26, 27)	26
1/11	$13^4, 15^3, 17^4$	$11, 13^4, 15^3, 17^3$	(36, 37)	36

Memory Elements m=3				
$d_{free} = 16$				
Period 3				
Rate	Encoding Matrices			Eff. Free Dist. Vector
k/n	$Encoder1$	$Encoder2$	$Encoder3$	$\tilde{d}_{eff}$
1/5	$13^2, 15, 17^2$	$11, 13, 15, 17^2$	$13^2, 15^2, 17$	(16, 16, 17)

Table 7: Time-Varying UEP Encoders

Memory Elements m = 4			
optimal $d_{free} = 7$			
Period 2			
Rate	**Encoding Matrices**		**Eff. Free Dist.**
k/n	$Encoder1$	$Encoder2$	$\tilde{d}_{eff}$
1/2	23, 35	31, 31	(5, 6)
1/2	23, 35	37, 31	(5, 7)
1/2	23, 35	33, 33	(6, 7)
1/2	23, 35	35, 37	(6, 8)
1/2	23, 35	37, 23	(7, 7)
1/2	23, 35	37, 25	(7, 8)
1/2	23, 35	27, 27	(7, 8)
1/2	23, 35	27, 35	(7, 8)
1/2	23, 35	35, 35	(7, 8)

Memory Elements m=7			
optimal$d_{free} = 10$			
Period 2			
Rate	**Encoding Matrices**		**Eff. Free Dist.**
k/n	$Encoder1$	$Encoder2$	$\tilde{d}_{eff}$
1/2	247, 371	247, 357	(8, 9)
1/2	247, 371	247, 373	(8, 10)
1/2	247, 371	247, 377	(8, 10)
1/2	247, 371	353, 333	(8, 12)
1/2	247, 371	331, 331	(9, 9)
1/2	247, 371	331, 353	(9, 10)
1/2	247, 371	331, 357	(9, 10)
1/2	247, 371	331, 333	(10, 10)
1/2	247, 371	247, 333	(10, 11)

Table 9: Time-varying UEP encoders

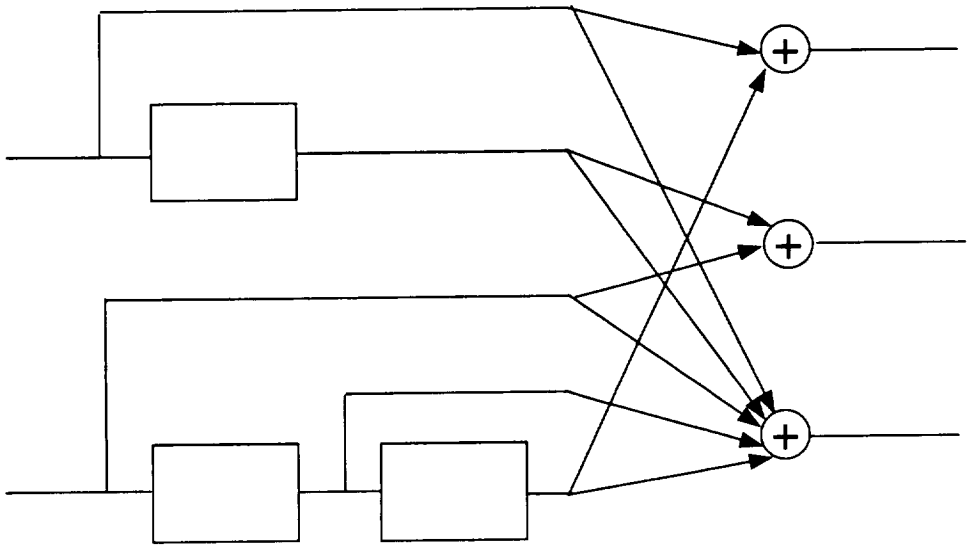

Figure 2: A Specific $(3, 2, 2)$ Convolutional encoder

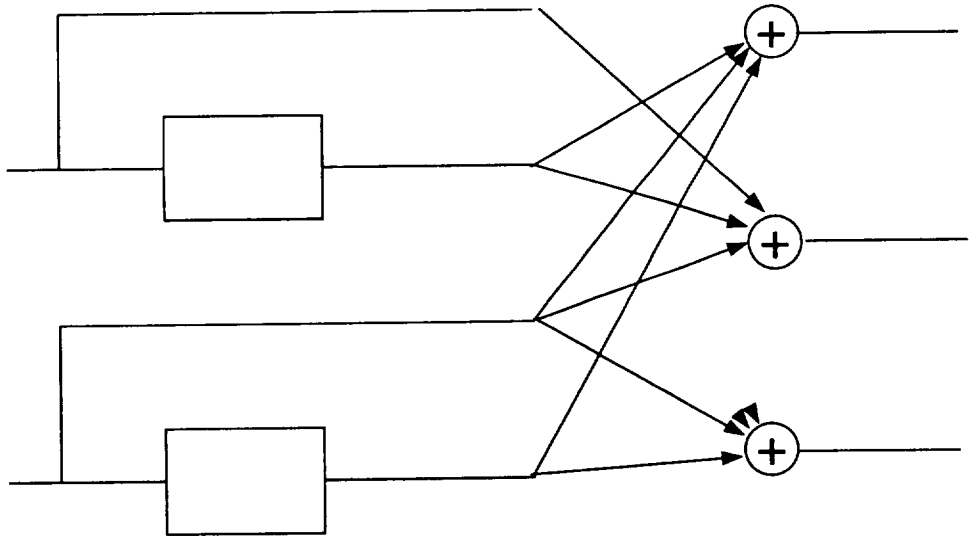

Figure 3: A specific $(3, 2)$ encoder

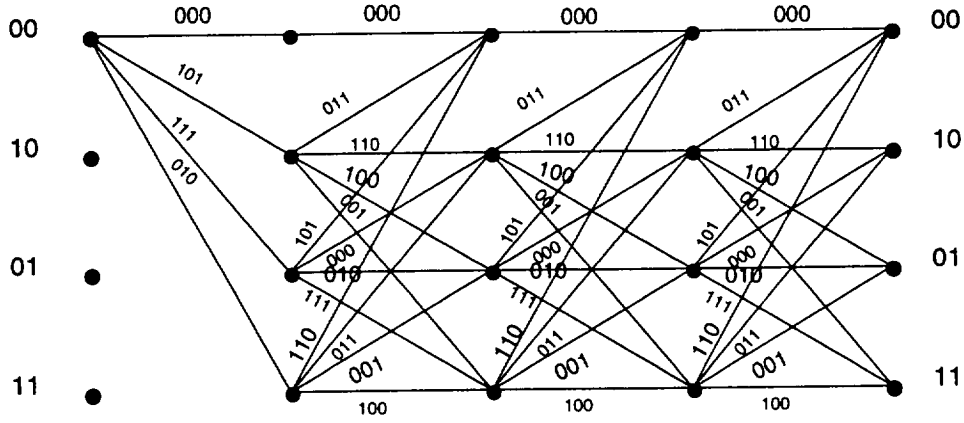

Figure 6: Trellis for a specific $(3, 2)$ encoder

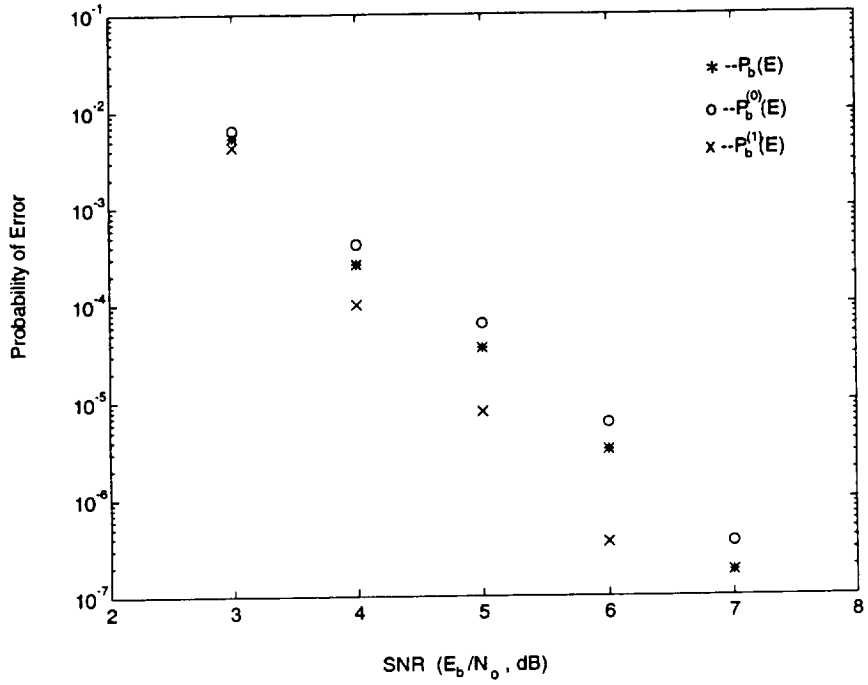

Figure 8: BER plot for $R = 2/3$, $\mathbf{M} = (2, 2)$ encoder with $\mathbf{d} = (4, 6)$

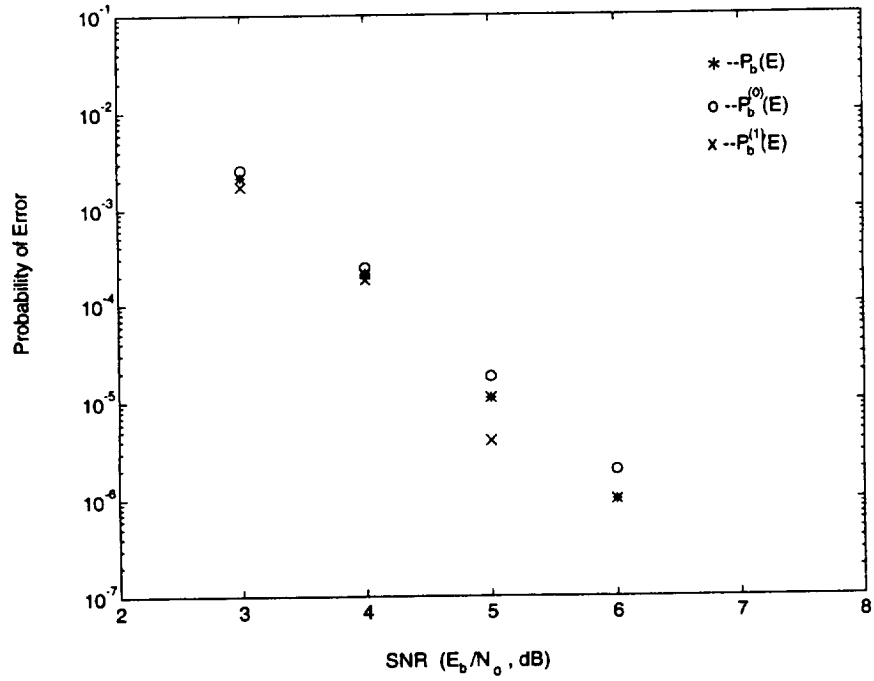

Figure 9: BER plot for $R = 2/4$, $\mathbf{M} = (1, 2)$ encoder with $\mathbf{d} = (6, 7)$