GENERATING SERIES AND PERFORMANCE BOUNDS FOR CONVOLUTIONAL CODES - PART 1: TRANSMISSION ON MEMORYLESS CHANNELS

Cecilio Pimentel

Communications Research Group - CODEC
Department of Electronics and Systems
Federal University of Pernambuco
P.O. Box 7800 - 50711-970 - Recife - PE
E.mail: cecilio@npd.ufpe.br

ABSTRACT

Performance bounds for convolutional codes over memoryless channels are commonly measured using the distance weight enumerator $T(x, y)$, also referred to as the *generating series* or the *transfer function*, of the code. In particular, bounds to the bit error probability are computed using the first few terms of the series expansion of $\{\partial T(x,y)/\partial x\}_{x=1}$. In this paper we present an efficient algebraic method to obtain this truncated series. We also propose a iterative procedure to compute the truncated $T(x, y)$ that discards, at each step, all paths with Hamming weight higher than a given order.

1. INTRODUCTION

A binary convolutional encoder is a linear finite state machine (FSM) that generates n_0 bits for every k_0 bits presented at its input. The rate of the encoder is $R_c = k_0/n_0$, and K is denoted the constraint length of the encoder. We will focus here on time invariant, non-catastrophic convolutional codes whose memory cells are arranged as a serial shift register. Figure 1 shows an example of a shift register convolutional encoder of rate $R_c = 1/2$, and constraint length $K = 4$. The state of the convolutional encoder is defined as the contents of the right most $(K - 1)\, k_0$ bits of the shift register.

The encoder state diagram is a labeled directed graph with $2^{(K-1)\, k_0}$ vertices, and $2^{K\, k_0}$ edges branches, each labeled with k_0-bit input and n_0-bit output strings. An extended state diagram that shows the evolution of state transitions at each time step is called a *trellis diagram*. Each possible transmitted codeword in a convolutional code corresponds to a unique path (sequence of states in the trellis diagram). The Hamming weight (number of 1's) of a path means the Hamming weight of the codeword associated to the path.

For the purpose of performance calculation, it is assumed,

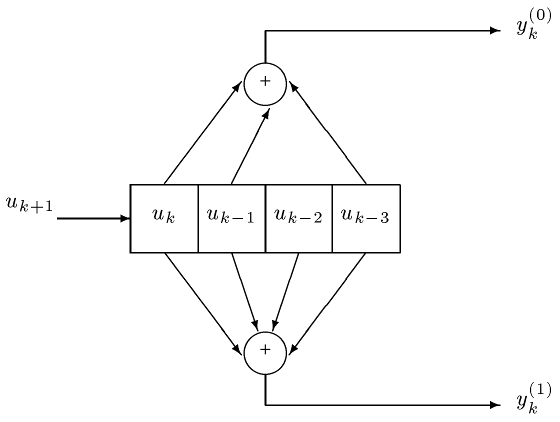

Figure 1: Encoder for a rate $R_c = 1/2$, constraint length $K = 4$, convolutional code.

without loss of generality, that the all-zero codeword (all zero path in the trellis diagram) is transmitted. Let $\mathcal{S}$ be the set of all paths that diverge from the all-zero path (leave the state 0), at a fixed time instant k, say $k = 0$, and remerge into the all-zero path exactly once at some time later. The performance analysis of convolutional codes is based on the *first-event error probability*, denoted here as P_{ev}, which is defined as the probability that any path in the set $\mathcal{S}$ accumulates higher metric than the all-zero path, given correct decoding up to $k = 0$. A second important performance measure is the *instantaneous bit error probability*, denoted as P_b, which is defined as the average number of erroneous information bits emerging from the decoder per information bit decoded. Using union bounds arguments, it is shown in [1, Section 4.4] that P_{ev} and P_b for memoryless channels are bounded above by:

$$\begin{aligned} P_{ev} &\leq \sum_{d=d_{free}}^{\infty} a_d\, P_d; \\ P_b &\leq \tfrac{1}{k_0} \sum_{d=d_{free}}^{\infty} b_d\, P_d. \end{aligned} \qquad (1)$$

In the expressions above, d_{free} is the minimum free distance of the code, a_d is the number of paths in $\mathcal{S}$ of Hamming weight d, b_d is the total number of nonzero information bits in all paths of Hamming weight d in $\mathcal{S}$. As for P_d, it is the pairwise probability that a path in $\mathcal{S}$ (a wrong path) of Hamming weight d is chosen instead of the correct path. The parameters a_d and b_d in Equation (1) depend only on the code parameters and are commonly calculated from the coder's transfer function $T(x,y)$ [1]. The transfer function enumerates the set $\mathcal{S}$ with respect to the Hamming weight of all input and output sequences of a convolutional code.

Typical methods for determining the transfer function include Mason's rule [2, 3], graph-reduction techniques [4], or solving state equations [1, 5]. The closed form expression for $T(x,y)$ rapidly becomes intractable as the number of states increases. For example, $T(x,y)$ for $K = 7$ is the ratio of two bivariate polynomials with 776 terms. In practice, however, the probabilities P_{ev} and P_b are calculated with acceptable accuracy using some initial terms of the series expansion of the transfer function. In this context, the complete $T(x,y)$ contains much more information than needed for performance calculation. In this paper, we propose a iterative procedure to compute the truncated $T(x,y)$ that discards, at each step, all paths with Hamming weight higher than a given order.

We adopt throughout this paper the following notation. The superscripts $\mathbf{A}^k$ and $\mathbf{A}^{-1}$ represent the k^{th} power and the inverse of the matrix $\mathbf{A}$, respectively. Moreover, $[\mathbf{A}]_{i,j}$ denotes the $(i,j)^{th}$ entry of $\mathbf{A}$. The matrix $\mathbf{I}$ stands for the identity matrix. If s and z are commutative indeterminates, $[s^k\, z^n]\, T(s,z)$ denotes the coefficient of $s^k\, z^n$ in the formal power series $T(s,z)$. $R[[x]]$ be the ring of all formal power series in commuting indeterminate x with coefficients taken from the field of real numbers R, and $R[x]$ is the set of all polynomials in x.

2. THE PATH WEIGHT ENUMERATORS

Let w_1 and w_2 be weight functions such that $w_1(\boldsymbol{\sigma})$ and $w_2(\boldsymbol{\sigma})$ are the number of 1's in the input and output (codeword) sequence, respectively, corresponding to an incorrect state sequence $\boldsymbol{\sigma} \in \mathcal{S}$. $T(x,y)$ is the generating series for the set $\mathcal{S}$ with respect to the weight functions w_1 and w_2, that is:

$$T(x,y) = \sum_{\boldsymbol{\sigma} \in \mathcal{S}} x^{w_1(\boldsymbol{\sigma})}\, y^{w_2(\boldsymbol{\sigma})} \qquad \in R[x][[y]]. \qquad (2)$$

$T(x,y)$ is a formal power series in y with a coefficient ring $R[x]$. It is clear that [1]:

$$\begin{aligned} a_d &= [y^d]\, T(1,y); \\ b_d &= [y^d] \left\{ \frac{\partial\, T(x,y)}{\partial x} \right\}_{x=1}. \end{aligned} \qquad (3)$$

In the following, we will derive an expression for $T(x,y)$ using the combinatorial method which enumerates sequences with restriction placed on adjacent symbols [6, Chapter 4],[7, Section 4.7]. The first step is to generate the adjacent matrix $\mathbf{A}$ of the encoder from the trellis diagram. For an encoder with 2^{K-1} states, $\mathbf{A}$ is a $2^{K-1} \times 2^{K-1}$ matrix whose $(i,j)^{th}$ entry is of the form $[\mathbf{A}]_{i,j} = x^{w_1(i \to j)}\, y^{w_2(i \to j)}$, where $w_1(i \to j)$ and $w_2(i \to j)$ are the Hamming weights of the input and output strings on the branch that connects the states i and j, respectively, for $0 \leq i,j \leq 2^{K-1} - 1$ (rows and columns of $\mathbf{A}$ are indexed by states). If states i and j are not joined on the trellis diagram, then $[\mathbf{A}]_{i,j}$ is set to zero. For example, the adjacent matrix for the convolutional code of Figure 1 is:

$$\mathbf{A} = \begin{bmatrix} 1 & xy^2 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & y^2 & x & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & y & xy & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & y & xy \\ y^2 & x & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & xy^2 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & y & xy & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & y & xy \end{bmatrix}. \qquad (4)$$

It is interesting to observe that for a shift-register convolutional code with constraint length K, all state sequences in $\mathcal{S}$ has the following structure: The first symbol is 0, the second is 1, the third is either 2 or 3, and so on, the second last symbol is 2^{K-2}, and the last symbol is 0. Define a non-zero path as a path which do not enter or leave the zero state. Let $T_1(x,y)$ be the generating series that enumerates non-zero paths from the initial state 1 to the terminal state 2^{K-2} with respect to the weight functions w_1 and w_2. An example of such a path for a convolutional code with $K = 4$ is $\boldsymbol{\sigma} = 1241241364$. Thus

$$T(x,y) = [\mathbf{A}]_{0,1}\, T_1(x,y)\, [\mathbf{A}]_{2^{K-2},0}. \qquad (5)$$

An expression for $T_1(x,y)$ is obtained directly from the adjacent matrix $\mathbf{A}$. Let $\mathbf{A}(0)$ be an adjacent matrix identical to its counterpart $\mathbf{A}$, except that the first row and the first column are set to zero, i.e., transitions from and into state 0 are not considered. Then, the generating series $T_1(x,y)$ is expressed as:

$$\begin{aligned} T_1(x,y) &= [(\mathbf{I} + \mathbf{A}(0) + \mathbf{A}(0)^2 + \mathbf{A}(0)^3 + \cdots)]_{1,2^{K-2}}; \\ &= \left[(\mathbf{I} - \mathbf{A}(0))^{-1}\right]_{1,2^{K-2}}. \end{aligned} \qquad (6)$$

The $(1, 2^{K-2})^{th}$-entry of the k^{th} power of $\mathbf{A}(0)$ is a bivariate polynomial in x and y whose exponents are Hamming weights $w_1(\boldsymbol{\sigma})$ and $w_2(\boldsymbol{\sigma})$, respectively, of all non-zero paths originating in state 1 and terminating in state 2^{K-2}, and the coefficients are the multiplicity of the weights. From Equations (6) and (5) we get [8]:

$$T(x,y) = [\mathbf{A}]_{0,1} \left[(\mathbf{I} - \mathbf{A}(0))^{-1} \right]_{1,2^{K-2}} [\mathbf{A}]_{2^{K-2},0}. \quad (7)$$

Notice that it is necessary to invert a $2^{K-1} \times 2^{K-1}$ symbolic matrix in order to find a closed form expression for $T(x,y)$. In particular, using the symbolic manipulation program MAPLE, we easily found $T(x,y)$ given by Equation (7) for the adjacent matrix $\mathbf{A}$ of Equation (4):

$$T(x,y) = \frac{(xy^2 - y - x)xy^6}{xy^3 + 2xy - 1} \quad \in R[x][[y]]. \quad (8)$$

The series expansion of Equation (8) yields the desired quantities a_d and b_d, since:

$$\begin{aligned} T(x,y) &= x^2y^6 + (x + 2x^3)y^7 + \cdots \\ T(1,y) &= y^6 + 3y^7 + 5y^8 + 11y^9 + \cdots \\ \left\{ \frac{\partial T(x,y)}{\partial x} \right\}_{x=1} &= 2y^6 + 7y^7 + 18y^8 + 49y^9 + \cdots \end{aligned} \quad (9)$$

Equation (8) shows $T(x,y)$ in a fractional form, which is converted into a truncated polynomial form in Equation (9) for the purpose of performance calculation. The computation of $T(x,y)$ is practical for relatively short constraint length. A method for finding the the series expansion of $T(1,y)$ and $\{\partial T(x,y)/\partial x\}_{x=1}$ for any finite number of terms without first computing the complete $T(x,y)$ is considered in the next section. Before we proceed we need the following definitions of equivalence of FSM's:

Definition 1: Two FSM's are said to be equivalent if and only if their transfer functions are identical.

Definition 2: Two FSM's are said to be equivalent of order L_m if and only if the series expansion of $T(x,y)$ and $\{\partial T(1,y)/\partial x\}_{x=1}$ of order L_m and lower are the same for the two FSM's, i.e., both FSM's have the same polynomials $\sum_{d=d_{free}}^{L_m} a_d\, y^d$ and $\sum_{d=d_{free}}^{L_m} b_d\, y^d$.

3. STATE REDUCTION ALGORITHM

An iterative procedure for calculating $T_1(x,y)$, called state reduction algorithm is developed in this section. The main idea is to create a sequence of adjacent matrices representing equivalent FSM's of order L_m with one state less.

It should be observed that each non-zero path is formed by concatenating paths that start from state 1 and reach state 2^{K-2} for the first time some time later. Call the set of all such paths $\mathcal{S}_2$. For example, the path $\boldsymbol{\sigma} = 1241241364$ is the concatenation of 3 paths, $\{124, 124, 1364\}$, belonging to $\mathcal{S}_2$. If $T_2(x,y)$ is the generating series for the set $\mathcal{S}_2$, we have:

$$\begin{aligned} T_1(x,y) &= T_2(x,y) + T_2(x,y)\,[\mathbf{A}]_{2^K-2,1}\,T_2(x,y) + \\ &\quad T_2(x,y)\,([\mathbf{A}]_{2^K-2,1}\,T_2(x,y)\,)^2 + \cdots \\ &= T_2(x,y)(1 - [\mathbf{A}]_{2^K-2,1}\,T_2(x,y))^{-1}. \end{aligned} \quad (10)$$

To calculate $T_2(x,y)$ we may form a sequence of equivalent FSM's where at each step we eliminate transitions from and into the r^{th} state. The $2^{K-1} \times 2^{K-1}$ adjacent matrix for this equivalent FSM, denoted by $\mathbf{A}(r)$, is calculated from the adjacent matrix of the previous step $\mathbf{A}(s)$ (obtained from the elimination of the s^{th} state) as shown in the following lemma.

Lemma 1 Let $\mathcal{R}$ and $\mathcal{C}$ be sets of indexes l, $l = 1, \cdots, 2^{K-1}$, $l \neq r$, such that $[\mathbf{A}(s)]_{l,r}$ and $[\mathbf{A}(s)]_{r,l}$ are different from zero, respectively. The $(i,j)^{th}$ entries of the matrix $\mathbf{A}(r)$ are:

$$\begin{aligned} &[\mathbf{A}(s)]_{i,j} + [\mathbf{A}(s)]_{i,r}(1 - [\mathbf{A}(s)]_{r,r})^{-1}[\mathbf{A}(s)]_{r,j}, \text{ if } i \in \mathcal{R},\ j \in \mathcal{C}; \\ &0, \quad \text{if } \ i = r, j = 1, \cdots, 2^{K-1}; \\ &0, \quad \text{if } \ j = r, i = 1, \cdots, 2^{K-1}; \\ &[\mathbf{A}(s)]_{i,j}, \quad \text{otherwise}, \end{aligned} \quad (11)$$

where on the first row, $[\mathbf{A}(s)]_{i,j}$ is due to parallel transitions, and the term $(1 - [\mathbf{A}(s)]_{r,r})^{-1}$ stands for the circulation loop on the r^{th} state. The state reduction algorithm is summarized below:

- Set $s = 0$. Find $\mathbf{A}(0)$.
- for $r = 2^{K-1} - 1, \cdots, 2^{K-2} - 1, 2^{K-2} + 1, \cdots, 2$ form the sequence of equivalent FSM's $\mathbf{A}(r)$ according to Lemma 1.
- $T_2(x,y) = [\mathbf{A}(2)]_{1,2^{K-2}}$.

In the following we demonstrate, as an example, the state reduction algorithm for the encoder of Figure 1 whose adjacent matrix is given by Equation (4).

- $s = 0, r = 7$:

$$\mathbf{A}(7) = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & y^2 & x & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & y & xy & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & \frac{y}{1-xy} & 0 \\ 0 & x & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & xy^2 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & y & xy & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}.$$

- $s = 7, r = 6$:

$$\mathbf{A}(6) = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & y^2 & x & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & y & xy & 0 & 0 \\ 0 & 0 & 0 & 0 & \frac{y^2}{1-xy} & \frac{xy^2}{1-xy} & 0 & 0 \\ 0 & x & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & xy^2 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}.$$

- $s = 6, r = 5$

$$\mathbf{A}(5) = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & y^2 & x & 0 & 0 & 0 & 0 \\ 0 & 0 & xy & x^2y^3 & y & 0 & 0 & 0 \\ 0 & 0 & \frac{xy^2}{1-xy} & \frac{x^2y^4}{1-xy} & \frac{y^2}{1-xy} & 0 & 0 & 0 \\ 0 & x & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}.$$

- $s = 5, r = 3$

$$\mathbf{A}(3) = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & \alpha & 0 & \frac{xy^2}{1-xy-x^2y^4} & 0 & 0 & 0 \\ 0 & 0 & \beta & 0 & \frac{y-xy^2}{1-xy-x^2y^4} & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & x & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix},$$

where $\alpha = (1-xy-x^2y^4+x^2)y^2/(1-xy-x^2y^4$, and $\beta = (xy-x^2y^2)/(1-xy-x^2y^4)$.

- $s = 3, r = 2$

$$\mathbf{A}(2) = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & \alpha & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & x & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix},$$

where $\alpha = (y+x-xy^2)y^2/1-2xy-x^2y^4+x^2y^2)$.

Then $[\mathbf{A}(2)]_{1,4} = T_2(x,y) = (y+x-xy^2)y^2/(1-2xy-x^2y^4+x^2y^2)$. Substituting $T_2(x,y)$ into Equation (10) and the result into Equation (5), we get $T(x,y)$ given by Equation (8), which is the same result as that obtained by other methods. Practical difficulties may arise for codes with moderate constraint length because the storage, but the algorithm is useful for finding a truncated transfer function using a symbolic manipulation program. We propose next a modification of the algorithm which is significant in practice. We will create a sequence of equivalent FSM's of order L_m (according to Definition 2) by performing the following operation: After calculating $[\mathbf{A}(r)]_{i,j},\ i \in \mathcal{R}, j \in \mathcal{C}$, according Lemma 1, we compute symbolically its series expansion with respect to the variable y, up to order L_m. As a result, each entry of $\mathbf{A}(r)$ is a bivariate polynomial of the form:

$$\sum_{i=0}^{L_m} p_i(x)\, y^i \quad \in R[x][[y]],$$

where $p_i(x)$ is a polynomial in x. In doing so, all paths with Hamming weight higher than L_m are discarded. The sequence of equivalent FSM's of order $L_m = 4$ is shown below:

- $s = 0, r = 7$:

$$\mathbf{A}(7) = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & y^2 & x & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & y & xy & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & \alpha & 0 \\ 0 & x & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & xy^2 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & y & xy & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix},$$

where $\alpha = y + xy^2 + x^2y^3 + x^3y^4$.

- $s = 7, r = 6$:

$$\mathbf{A}(6) = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & y^2 & x & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & y & xy & 0 & 0 \\ 0 & 0 & 0 & 0 & \alpha & \beta & 0 & 0 \\ 0 & x & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & xy^2 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix},$$

where $\alpha = y^2 + xy^3 + x^2y^4$ and $\beta = xy^2 + x^2y^3 + x^3y^4$

- $s = 6, r = 5$

$$\mathbf{A}(5) = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & y^2 & x & 0 & 0 & 0 & 0 \\ 0 & 0 & xy & x^2y^3 & y & 0 & 0 & 0 \\ 0 & 0 & \alpha & x^2y^4 & \beta & 0 & 0 & 0 \\ 0 & x & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix},$$

$\alpha = xy^2 + x^2y^3 + x^3y^4$, and $\beta = y^2 + xy^3 + x^2y^4$.

- $s = 5, r = 3$

$$\mathbf{A}(3) = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & \alpha & 0 & \beta & 0 & 0 & 0 \\ 0 & 0 & xy & 0 & y & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & x & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix},$$

where $\alpha = (1 + x^2)y^2 + x^3y^3 + x^4y^4$, and $\beta = xy^2 + x^2y^3 + x^3y^4$.

- $s = 3, r = 2$

$$\mathbf{A}(2) = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & \alpha & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & x & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix},$$

where $\alpha = xy^2 + (1 + 2x^2)y^3 + (x + 3x^3)y^4$.

The truncated transfer function $T_2(x, y)$ is $xy^2+(1+2x^2)y^3+(x + 3x^3)y^4$. From this polynomial we can easily calculate the truncated series $T_1(x, y)$ and the truncated $T(x, y)$ which is equivalent of order L_m to the complete $T(x, y)$. The main feature of the modified algorithm is that all operations are performed in the coefficient ring $R[x][[y]]/(y^{L_m})$. Notice that no matter the number of states, the entries of $\mathbf{A}(r)$ are bivariate polynomials whose powers of y are of order at most L_m.

Example 3.1 *The truncated generating series up to order 15 for the 1/2-rate convolutional code with generator polynomial (in octal) 4734,6624 is:*

$$\begin{aligned} T(x, y) = \;& (x^4 + x^{10})y^{12} + (x + 5x^3 + 2x^5)y^{13}+ \\ & (3x^2 + 3x^4 + 8x^6 + x^8)y^{14}+ \\ & (3x^3 + 8x^5 + 11x^7 + 6x^9 + 7x^{11})y^{15}. \end{aligned} \quad (12)$$

The coefficients $\{b_d\}_{d=d_{free}}^{22}$ for 1/2-rate convolutional codes are shown in table 1.

4. COMMENTS

Signal flow graphs [9] is the well known technique employed to construct equivalent FSM's with one state less and is largely used in conjunction with Mason's rules to find the transfer function of convolutional codes. There is an analogy between the signal flow graph rules and combinatorial principles that enumerates sequences with restrictions on adjacent symbols. This work presents an algorithm to compute transfer functions with two new features. First, we defined combinatorial identities to work with equivalent FSM's at the level of the adjacent matrix which is convenient for symbolic computation. Second, operations are performed in the coefficient ring, resulting in a truncated transfer function with considerable less storage requirements.

Table 1: Weight coefficients of 1/2-rate convolutional codes.

$K =$	7	8	9	10
Generators	133,171	247,371	753,561	4734,6624
b_8	-	-	-	-
b_9	-	-	-	-
b_{10}	36	2	-	-
b_{11}	0	22	-	-
b_{12}	211	60	33	14
b_{13}	0	148	0	26
b_{14}	1404	340	218	74
b_{15}	0	1008	0	256
b_{16}	11633	2642	2179	496
b_{17}	0	6748	0	1378
b_{18}	77433	18312	15035	4122
b_{19}	0	48478	0	10832
b_{20}	502690	126364	105166	27988
b_{21}	0	320062	0	72209
b_{22}	3322763	821350	692330	186920

5. ACKNOWLEDGMENTS

The author acknowledges partial support from the Brazilian National Council for Scientific and Technological Development (CNPq) under Grant No. 300987/96-0.

6. REFERENCES

[1] Andrew J. Viterbi and Jim K. Omura, *Principles of Digital Communication and Coding,* McGraw-Hill Book Company, New York, 1979.

[2] Stephen B. Wicker, *Error Control Systems for Digital Communication and Storage,* Prentice-Hall Inc., New Jersey, 1995.

[3] Shu Lin and Daniel J. Costello, *Error Control Coding,* Prentice-Hall, Inc., Englewood Cliffs, USA, 1983.

[4] Robert J. McEliece, *The Theory of Information and Coding,* Addison-Wesley Reading, MA, 1977.

[5] John G. Proakis, *Digital Communications*, McGraw-Hill Book Company, Third Edition, New York - USA, 1995.

[6] I. P. Goulden and D. M. Jackson, *Combinatorial Enumeration*, John Wiley & Sons, 1983.

[7] R. P. Stanley, *Enumerative Combinatorics*, vol. I. Monterey: Wadsworth & Brooks/Cole, 1986.

[8] Cecilio Pimentel, Enumeration Techniques for Finite State Channels, PhD Thesis, University of Waterloo, July, 1996.

[9] S. Mason and H. Zimmermann, *Electronic Circuits, Signals, and Systems*, New York: Wiley, 1960.