Using the Fourier Transform to Compute the Weight Distribution of a Binary Linear Block Code

Pragat Chaudhari and Amir K. Khandani

***Abstract*—An analytical technique is presented to compute the weight distribution of a linear block code by performing a Fourier analysis involving certain matrices obtained from the code trellis. The proposed method is general, easy to implement, and can be used without having to traverse the trellis or carry out tedious analytical work. The introduced technique can be used as a flexible analytical tool to capture the weight structure of the code with application to problems involving analysis and/or design.**

***Index Terms*—Binary linear block code, discrete Fourier transform, state transition matrix, turbo code, weight distribution.**

I. Introduction

KNOWLEDGE of the weight distribution of linear codes is important in their error performance analysis. Due to this fact, numerous research works have addressed the problem of computing the weight distribution of general or specific code constructions.

The techniques known for computing the weight distribution of a general linear code are based on representing the code by a state diagram in the case of convolutional codes [2], [3], or by a trellis diagram[1] in the case of block codes [4]–[9]. These methods are based on assigning a partial weight enumeration function to the transitions of a state (or trellis) diagram, where the partial weight distributions are appropriately multiplied and summed (reflecting the concept of state in traversing the allowed paths) to yield the complete weight distribution of the code. Similar computational techniques have been used in conjunction with constrained coding systems as well [10].

The focus of this paper is to present the use of a variation of the discrete Fourier transform (DFT) operating on matrices to calculate the weight distribution of a linear block code. This is achieved using a modified state transition matrix which is defined using the basic concepts of the DFT. The corresponding computation involves raising the modified state transition matrix to the power of the code block length and applying the inverse DFT to the result. The proposed method is general, easy to implement, and unlike other known methods, does not require traversing the trellis or performing tedious analytical work. The introduced technique provides a flexible and insightful analytical tool to capture the weight structure of the code with application to a variety of problems involving analysis and/or design.

The article is organized as follows: Section II presents the formation of the modified state transition matrix, called the *weighted state transition matrix.* The use of the Fourier analysis is presented, allowing us to compute the weight distribution of the code in a systematic manner. Section III is devoted to simple examples to illustrate the main procedure. Finally, Section IV contains a brief summary of the article.

II. State Transitions Matrices

Consider a trellis $\mathcal{T}$ with K states, $s_0, \ldots, s_{K-1}$ where each transition between a pair of states (s_i, s_j) is distinguished by one or several input bit(s), as well as one or several output bit(s). We define the partial state transition matrices of $\mathcal{T}$ as a set of $K \times K$ matrices $\mathbf{T}^{(k)}_{m,n}$ where the (i, j)th element of $\mathbf{T}^{(k)}_{m,n}$, namely $T^{(k)}_{m,n}(i, j)$, is equal to the number of transitions of input weight m and output weight n between states i and j after k transitions. We will consider $\mathbf{T}^{(k)}_{m,n}$, where $m, n = 0, 1, \ldots$, as two-dimensional discrete series elements. It is shown that the calculation of weight distribution reduces to computing certain convolutions defined on the sequence of such matrices. This is subsequently converted into multiplication by defining an appropriate form of Fourier transform pair. To show this, we note that the number of paths with input weight m and output weight n between states i and j after k transitions through a given trellis, can be found using the following recursive relationship:

$$T^{(k)}_{m,n}(i, j) = \sum_p \sum_q \sum_\alpha T^{(k-1)}_{m-p,n-q}(i, \alpha)\, T^{(1)}_{p,q}(\alpha, j). \quad (1)$$

In matrix form, we have

$$\mathbf{T}^{(k)}_{m,n} = \sum_p \sum_q \mathbf{T}^{(k-1)}_{m-p,n-q} \mathbf{T}^{(1)}_{p,q}. \quad (2)$$

This is indeed the convolution of the sequences $\mathbf{T}^{(k-1)}_{m,n}$ and $\mathbf{T}^{(1)}_{m,n}$. We define the weighted state transition matrix $\mathbf{X}(u, v)$ as

$$\mathbf{X}(u, v) = \sum_{m=0}^{L_1-1} \sum_{n=0}^{L_2-1} \mathbf{T}^{(1)}_{m,n} U^{mu} V^{nv} \quad (3)$$

where

$$U = \exp\left(-\frac{j2\pi}{L_1}\right), \quad V = \exp\left(-\frac{j2\pi}{L_2}\right), \quad j = \sqrt{-1} \quad (4)$$

Manuscript received March 3, 2000. The associate editor coordinating the review of this letter and approving it for publication was Dr. M. Fossorier. This work was supported in part by the Natural Sciences and Engineering Research Council of Canada (NSERC) and by Communications and Information Technology Ontario (CITO). A more detailed version of this work can be found in [1].

The authors are with the University of Waterloo, Waterloo, ON, Canada (e-mail: pragat@shannon.uwaterloo.ca; khandani@shannon.uwaterloo.ca).

Publisher Item Identifier S 1089-7798(01)01486-7.

[1] A trellis diagram differs from a state diagram in that a time axis is associated with the transitions.

and L_1, L_2 are selected as arbitrary integers larger than the maximum possible input weight and maximum possible output weight, respectively, to avoid aliasing in the subsequent inverse transform operation. Note that $\mathbf{T}^{(1)}_{m,n}$ and $\mathbf{X}(u, v)$ form a DFT pair, i.e.,

$$\mathbf{T}^{(1)}_{m,n} \overset{\mathcal{F}}{\longleftrightarrow} \mathbf{X}(u, v) \tag{5}$$

where $\mathcal{F}$ denotes the discrete Fourier transform operation as defined in our context. Using basic orthogonality properties of Fourier transform pairs, it is easy to show that the discrete Fourier transform of the convolutional operation in (2) yields a product of the DFT's of the two transitions matrices. Furthermore, by recursively applying this property, it can be established that

$$\mathbf{T}^{(N)}_{m,n} \overset{\mathcal{F}}{\longleftrightarrow} \mathbf{X}^N(u, v). \tag{6}$$

Using this property, to compute the weight enumeration function over N consecutive stages of the trellis, we simply raise the matrix $\mathbf{X}(u, v)$ to the power N (to encapsulate that N transitions have occurred) and then perform the inverse transform, resulting in

$$\mathbf{A}_{m,n} = \frac{1}{L_1 L_2} \sum_{u=0}^{L_1-1} \sum_{v=0}^{L_2-1} \mathbf{X}^N(u, v) U^{-mu} V^{-nv}. \tag{7}$$

The (i, j)th element of the $K \times K$ matrix $\mathbf{A}_{m,n}$ indicates the number of paths of input weight m and output weight n starting at state i and ending at state j after traversing N consecutive stages of the trellis. The main computational step in computing (7) is to raise the $K \times K$ matrix $\mathbf{X}(u, v)$ to the power of N. This can be achieved easily by using an eigenvalue decomposition of $\mathbf{X}(u, v)$ and raising the eigenvalues to the power of N [11].

In general, the entries of matrix $\mathbf{A}_{m,n}$ have an exponential growth with N. As a result, for large values of N one may encounter numerical difficulties in using (7). This problem can be easily handled by performing the calculations on shorter sub-blocks, truncating the resulting partial weight distributions, combining the results through multiplication of the corresponding weighted state transition matrices, and finally performing the inverse Fourier operation on the result. Note that similar precautions are needed in any other method used to compute the weight distribution.

For a linear code, it is required that the trellis begins and ends in the "zero" state. This corresponds to location (0, 0) of the matrix $\mathbf{A}_{m,n}$. The other entries of the matrix provide the weight distributions of the cosets of this linear code.

We usually have $L_1 = L_2$ resulting in $U = V$ in which case we use L to represent the common value of $L_1 = L_2$ and W to represent the common value of $U = V$.

The above formulation accounts for the contributions of the input and output weights, separately. In some situations, we may be interested only in the weight of the output, in which case we can omit the variable u in (3) and (7), and express the Fourier transform pair in terms of a single summation, as in the pair of equations below.

$$\mathbf{X}(v) = \sum_{n=0}^{L-1} \mathbf{T}^{(1)}_n W^{nv} \tag{8}$$

$$\mathbf{A}_n = \frac{1}{L} \sum_{v=0}^{L-1} \mathbf{X}^N(v) W^{-nv}. \tag{9}$$

The developed methodology is quite versatile and can be applied to any code which is representable by a trellis diagram, including convolutional codes, Turbo codes, and many other linear block codes. For the case of Turbo codes, the coefficients of conditional weight enumeration functions (as defined in [5]) can be easily calculated for the error performance analysis.

The memory requirement for the proposed method is in the order of K^2, while the memory requirements of conventional methods based on traversing the trellis are in the order of $M_I M_O K$, where M_I, M_O are the maximum weights of the input and output, respectively. Note that for a code with a reasonable state complexity, usually $M_I M_O \gg K$, meaning that the proposed method has a smaller memory requirement.

For a code with a large number of states the proposed method will be too complex to implement, however, for such a large code, any other technique will also be complex. In general, for such a large code, one needs to somehow use the specific code construction to simplify the calculations, either in conjunction with the method proposed here, or in conjunction with any other alternative method including those based on a trellis representation.

Computational complexity of the proposed method is not of a major concern and is generally comparable to other alternative techniques known for this purpose. In general, note that the main objective of the proposed method is to provide for an easily implementable algorithm. This means that the complexity issues are not the main focus of the discussion.

The validity of the proposed method has been verified by comparing the resulting weight distributions for various codes to those obtained by a direct computation of (1) using the trellis diagram.

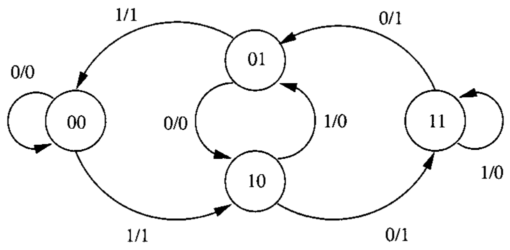
III. Some Examples

A. A Recursive Convolutional Code

Consider the simple $(5, 7)_8$ recursive convolutional code, where 5_8 represents the taps on the memory elements for the output bits, and 7_8 represents the feedback taps. This is a 2 memory element code, with 4 states. The state diagram is as shown below in Fig. 1. From the state diagram, we have,

$$\mathbf{T}^{(1)}_{0,0} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix}, \quad \mathbf{T}^{(1)}_{0,1} = \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 1 & 0 & 0 \end{bmatrix},$$

$$\mathbf{T}^{(1)}_{1,0} = \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}, \quad \mathbf{T}^{(1)}_{1,1} = \begin{bmatrix} 0 & 0 & 1 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix}. \tag{10}$$



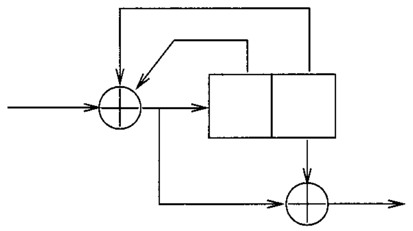


Fig. 1. State diagram of $(5, 7)_8$ recursive convolutional code.

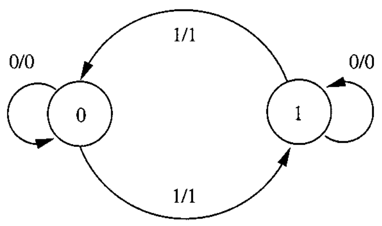


Fig. 2. State diagram of a generic single-parity check code (valid code sequences begin and end in the zero state).

Using (3), we obtain,

$$\mathbf{X}(u, v) = \begin{bmatrix} 1 & 0 & W^{u+v} & 0 \\ W^{u+v} & 0 & 1 & 0 \\ 0 & W^u & 0 & W^v \\ 0 & W^v & 0 & W^u \end{bmatrix},$$

where

$$W = \exp\left(-\frac{j2\pi}{L}\right), L > N. \qquad (11)$$

Using (7) and (11), the weight distribution of the code can be found.

B. Single-Parity Check Codes

Consider a simple $(N, N-1)$ single-parity check code. The state diagram of the code is provided in Fig. 2. From this state diagram and using (8), we obtain the following weighted state transition matrix (considering the output weights only):

$$\mathbf{X}(v) = \begin{bmatrix} 1 & W^v \\ W^v & 1 \end{bmatrix}, \qquad W = \exp\left(-\frac{j2\pi}{L}\right), L > N. \qquad (12)$$

In (9), the coefficient is calculated by raising $\mathbf{X}(v)$ to the Nth power and the inverse DFT is then performed. The eigenvalues of $\mathbf{X}(v)$ with corresponding eigenvectors, can be verified to be,

$$\lambda_1 = 1 - W^v \quad p_1 = [1 \quad -1]^T \qquad (13)$$
$$\lambda_2 = 1 + W^v \quad p_2 = [1 \quad 1]^T. \qquad (14)$$

Using these values, $\mathbf{X}^N$ can be calculated as

$$\mathbf{X}^N = \mathbf{P}\mathbf{\Lambda}^N\mathbf{P}^{-1} \qquad (15)$$

where, $\mathbf{P} = [p_1 \ p_2]$ and $\mathbf{\Lambda} = \text{diag}(\lambda_1, \lambda_2)$ Using (9), the coefficients can be calculated for different n values. Implementing this procedure for the (5, 4) single-parity check code, the weight enumeration below is obtained.

Weight	Weight Coeffcient, A_n
0	1
2	10
4	5

IV. Summary

A systematic method is presented to calculate the weight distribution of a linear block code expressed in terms of its trellis structure. The proposed method is general, easy to implement, and can be used without having to traverse the trellis or carry out tedious analytical work.

References

[1] P. Chaudhari, "Analytical methods for the performance evaluation of binary linear block codes," M.A.Sc. thesis, Dep. Elect. Comp. Eng., Univ. Waterloo, Canada, 2000.

[2] S. J. Mason, "Feedback theory—Further properties of signal flow graphs," *Proc. IRE*, vol. 44, pp. 920–926, July 1956.

[3] S. J. Mason and H. J. Zimmerman, *Electron. Circuits, Signals, Syst.*. New York, NY: Wiley, 1960.

[4] J. K. Wolf and A. J. Viterbi, "On the weight distribution of linear block codes formed from convolutional codes," *IEEE Trans. Commun.*, vol. IT-44, pp. 1049–1051, Sept. 1996.

[5] S. Benedetto and G. Montorsi, "Unveiling turbo codes: Some results on parallel concatenated coding schemes," *IEEE Trans. Inform. Theory*, vol. IT-42, pp. 409–428, Mar. 1996.

[6] Y. Desaki, T. Fujiwara, and T. Kasami, "A method for computing the weight distribution of a block code by using its trellis diagram," *IEICE Trans. Fundamentals*, vol. E77-A, pp. 1230–1237, Aug. 1994.

[7] ——, "The weight distributions of extended binary primitive BCH codes of length 128," *IEEE Trans. Inform. Theory*, vol. IT-43, pp. 1364–1371, July 1997.

[8] O. T. Takeshita, M. P. C. Fossorier, and D. J. Costello, "A new technique for computing the weight spectrum of turbo-codes," *IEEE Commun. Lett.*, vol. 3, pp. 251–253, Aug. 1999.

[9] M. P. C. Fossorier, S. Lin, and D. J. Costello, "On the weight distribution of terminated convolutional codes," *IEEE Trans. Inform. Theory*, vol. IT-45, pp. 1646–1648, July 1999.

[10] R. E. Blahut, *Principles and Practice of Information Theory*. Reading, MA: Addison-Wesley, 1987.

[11] R. A. Horn and C. R. Johnson, *Topics in Matrix Analysis*. Cambridge, U.K.: Cambridge Univ. Press, 1993.