Numerical Calculation Algorithm for Transfer Function Bounds of Convolutional Codes and TCM Schemes
畳み込み符号およびTCM方式における伝達関数を用いた誤り上界式に対する数値計算アルゴリズム

Hirosuke Yamamoto
山本博資

Hiroyuki Fujiwara
藤原弘之

Department of Communications and Systems
University of Electro-Communications
電気通信大学電子情報学科

ABSTRACT A new algorithm is proposed to calculate the transfer function $T(D)$ and $\frac{\partial T(D,I)}{\partial I}|_{I=1}$ numerically, which are included in the well-known upper bounds of the error-event and bit-error probabilities, respectively, for convolutional codes and trellis-coded modulation schemes. The algorithm is an iteration method which is based on the power method to approximate an eigenvector of the dominant eigenvalue of a matrix. The differential $\frac{\partial T(D,I)}{\partial I}|_{I=1}$ is calculated without any difference operation.

I. INTRODUCTION

For convolutional codes and trellis coded modulation (TCM) schemes, the theoretical bounds of the event-error or bit-error probabilities can be represented by the transfer function (or generating function) of the encoder state diagram. (For more details, see [1] for convolutional codes and [2] for TCM schemes[1].) For the sake of illustration, we consider a simple convolutional encoder given in Fig. 1. The state diagram of this encoder is shown in Fig. 2 and the transfer function $T(D,I)$ is given by

$$T(D,I) = \frac{D^5 I}{1-2DI} = D^5 I + 2D^6 I^2 + 2^2 D^7 I^3 + \cdots + 2^k D^{5+k} I^{1+k} + \cdots. \tag{1}$$

The event-error probability P_e and the bit-error probability P_b for rate b/n convolutional codes are bounded above as follows.

$$P_e \le T(D,I)|_{I=1,D=Z}, \qquad P_b \le \frac{1}{b}\left.\frac{\partial T(D,I)}{\partial I}\right|_{I=1,D=Z}, \tag{2}$$

where Z depends on channel noise. For the additive white Gaussian channel, these bounds are tightened as follows.

$$P_e \le Q\left(\sqrt{\frac{2d_f\mathcal{E}_s}{N_0}}\right) e^{d_f\mathcal{E}_s/N_0} T(D,I)|_{I=1,D=e^{-\mathcal{E}_s/N_0}},$$

$$P_b \le \frac{1}{b}Q\left(\sqrt{\frac{2d_f\mathcal{E}_s}{N_0}}\right) e^{d_f\mathcal{E}_s/N_0} \left.\frac{\partial T(D,I)}{\partial I}\right|_{I=1,D=e^{-\mathcal{E}_s/N_0}},$$

[1] In this paper, we use the same notation as [1].

where $\mathcal{E}_s/N_0$ is SN ratio per source symbol.

Since the number of states increases exponentially as the constraint length of a convolutional code becomes long, it is hard to treat the transfer function symbolically. Hence the above bounds are usually calculated numerically. In the numerical computation, the partial derivative at $I = 1$ is approximated as

$$\left.\frac{\partial T(D,I)}{\partial I}\right|_{I=1,D=Z} \lesssim \frac{T(Z,1+\varepsilon) - T(Z,1)}{\varepsilon}, \quad \varepsilon \ll 1. \tag{3}$$

On the other hand, $T(D) \triangleq T(D,I)|_{I=1}$ can be obtained by solving the following state equations.

$$\begin{cases} \xi_b &= D^2 + \xi_c \\ \xi_c &= D\xi_b + D\xi_d \\ \xi_d &= D\xi_b + D\xi_d \end{cases} \tag{4}$$

$$T(D) = D^2\xi_c \tag{5}$$

where ξ's represent the value of each internal node for the unity input in Fig. 2. Besides solving the above equations directly, it is known that the solution can be obtained numerically by using the series expansion of an inverse matrix [1, Prob. 4.18]. Let $\boldsymbol{\xi} \triangleq (\xi_b, \xi_c, \xi_d)$, $\mathbf{b} \triangleq (D^2, 0, 0)$, and

$$A \triangleq \begin{bmatrix} 0 & D & D \\ 1 & 0 & 0 \\ 0 & D & D \end{bmatrix} \tag{6}$$

Then, since (4) can be represented as $\boldsymbol{\xi} = \boldsymbol{\xi}A + \mathbf{b}$, we have

$$\begin{aligned} \boldsymbol{\xi} &= \mathbf{b}(I-A)^{-1} = \mathbf{b}(I + A + A^2 + A^3 + \cdots) \\ &= \sum_{n=\ell}^{\infty} \mathbf{S}^{(\ell)}, \end{aligned} \tag{7}$$

where $\mathbf{S}^{(\ell)}$ is defined as

$$\mathbf{S}^{(\ell)} = \begin{cases} \mathbf{b}, & \text{if } \ell = 0 \\ \mathbf{S}^{(\ell-1)}A, & \ell = 1, 2, \cdots. \end{cases} \tag{8}$$

By truncating the summation of (7) properly, we can approximate the solution with adequate accuracy. It is shown in [3] that the calculation time can be reduced by the series expansion method compared to the direct calculation of (4).

The above series expansion method can be simplified by including the start node $\xi_a = 1$ of the state diagram into $\boldsymbol{\xi}$. Redefine the vector $\boldsymbol{\xi}$ as $\boldsymbol{\xi} \triangleq (\xi_a, \xi_b, \xi_c, \xi_d)$ and let

$$B \triangleq \begin{bmatrix} 1 & D^2 & 0 & 0 \\ 0 & 0 & D & D \\ 0 & 1 & 0 & 0 \\ 0 & 0 & D & D \end{bmatrix}. \tag{9}$$

Then (4) can be represented as

$$\boldsymbol{\xi} = \boldsymbol{\xi}B. \tag{10}$$

This equation means that $\boldsymbol{\xi}$ is an eigenvector for the eigenvalue $\lambda = 1$ of B. (Note from (9) that B has $\lambda = 1$.) Other eigenvalues depend on D. However, we can easily show that if the absolute value of an eigenvalue is greater than one, $T(D)$ becomes infinity. Hence, for the case we are interested in, $\lambda = 1$ is the dominant eigenvalue of B and we can use the power method to get the eigenvector $\boldsymbol{\xi}$ for $\lambda = 1$ as follows. (See e.g. [4, Sec. 8.4] for the details of the power method.)

Power Method for $T(D)$

1. (*Initialization*)
   $\boldsymbol{\xi}^{(0)} := (1,0,0,0)$, $\ell := 0$.
2. **repeat**
   (a) $\ell := \ell + 1$.
   (b) $\boldsymbol{\xi}^{(\ell)} := \boldsymbol{\xi}^{(\ell-1)}B$.
   until sufficient accuracy is achieved.
3. $T(D) := D^2\xi_c$

Since matrix B becomes more sparse as the constraint length of the convolutional codes becomes longer, the above algorithm can be speeded up by implementing the calculation $\boldsymbol{\xi}^{(\ell-1)}B$ element-wisely. In the next section, we will establish such algorithm to calculate $T(D)$. Furthermore, by modifying the algorithm, we will show in section 3 that $\frac{\partial T(D,I)}{\partial I}|_{I=1}$ can be calculated without any difference operation.

II. ALGORITHM TO CALCULATE $T(D)$

For simplicity, we consider rate $1/n$ binary convolutional codes with constraint length K. But the following algorithm can be easily generalized to any b/n convolutional codes or any TCM schemes.

Let $u_{K-1}u_{K-2}\cdots u_2u_1$ ($u_k \in \{0,1\}, k = 1\cdots K-1$) be contents of the $K-1$ bits shift register of a convolutional encoder and let u_0 be a new coming information bit. If the generator matrix G of the encoder is

$$G = \begin{bmatrix} g_{0,1} & g_{0,2} & \cdots & g_{0,n} \\ g_{1,1} & g_{1,2} & \cdots & g_{1,n} \\ \vdots & \vdots & & \vdots \\ g_{K-1,1} & g_{K-1,2} & \cdots & g_{K-1,n} \end{bmatrix}, \tag{11}$$

then n encoder output bits, say $v_1, v_2, \cdots, v_n$, are determined by

$$v_j = \sum_{k=0}^{K-1} u_k g_{k,j}, \quad j = 1,2,\cdots,n. \tag{12}$$

Furthermore the Hamming weight of the output bits $(v_1, v_2, \cdots, v_n)$ can be obtained by

$$\begin{aligned} w(u_{K-1}u_{K-2}\cdots u_1u_0) &\triangleq v_1 + v_2 + \cdots + v_n \\ &= \sum_{j=1}^{n}\sum_{k=0}^{K-1} u_k g_{k,j}. \end{aligned} \tag{13}$$

For this code, we represent each node in the state diagram as $\xi_{u_{K-1}u_{K-2}\cdots u_1}$. Since $u_{K-1}u_{K-2}\cdots u_1$ moves to $u_{K-2}\cdots u_1u_0$ at the next clock, $\xi_{u_{K-1}u_{K-2}\cdots u_1}$ is connected to $\xi_{u_{K-2}\cdots u_1 0}$ and $\xi_{u_{K-2}\cdots u_1 1}$ in the state diagram, or in other words, $\xi_{u_{K-2}u_{K-3}\cdots u_0}$ is connected from $\xi_{0u_{K-2}u_{K-3}\cdots u_1}$ and $\xi_{1u_{K-2}u_{K-3}\cdots u_1}$. Let D^S be the label of a branch in the state diagram, which goes from $\boldsymbol{\xi}_{u_{K-1}u_{K-2}\cdots u_1}$ to $\boldsymbol{\xi}_{u_{K-2}\cdots u_1u_0}$. Since the superscript S of D^S stands for the weight of the encoder output bits, S is equal to $w(u_{K-1}u_{K-2}\cdots u_1u_0)$. Hence, the power method algorithm shown in section 1 can be described as follows.

Element-wise Power Method for $T(D)$

1. (*Initialization*)
   $\xi^{(0)}_{0\cdots0} := 1$ and let other $\xi^{(0)}_{u_{K-1}u_{K-2}\cdots u_1}$'s be zero.
   $\ell := 0$.

2. **repeat**

   (a) $\ell := \ell + 1$.

   (b) $\xi^{(\ell)}_{0\cdots0} := \xi^{(\ell-1)}_{0\cdots0}$.

   (c) **for** $i := 1$ **to** $2^{K-1} - 1$ **do**

      i. $u_{K-2}u_{K-3}\cdots u_0 := (i)_2^{K-1}$.
      ($(i)_2^{K-1}$ stands for the binary representation of i with $K-1$ bits.)

      ii.

$$\xi^{(\ell)}_{u_{K-2}u_{K-3}\cdots u_0} := D^{w(0u_{K-2}u_{K-3}\cdots u_1)}\xi^{(\ell-1)}_{0u_{K-2}u_{K-3}\cdots u_1} + D^{w(1u_{K-2}u_{K-3}\cdots u_1)}\xi^{(\ell-1)}_{1u_{K-2}u_{K-3}\cdots u_1}. \quad (14)$$

   until sufficient accuracy is achieved.

3. $T(D) := D^{w(100\cdots0)}\xi_{100\cdots0}$.

This algorithm can be programmed without considering the state diagram if the generator matrix G to calculate $w(u_{K-1}\cdots u_0)$ is given. Hence, we can easily write a general program, by which we can calculate $T(D)$ for any $1/n$ convolutional code.

In the above algorithm, $\boldsymbol{\xi}$ converges rapidly if D is much less than one. But, the convergence speed can further be improved by modifying the algorithm slightly. In (14), $\xi^{(\ell)}_{u_{K-2}u_{K-3}\cdots u_0}$ is updated by using $\xi^{(\ell-1)}_{u_{K-1}u_{K-2}\cdots u_1}$. However, if $\xi^{(\ell)}_{u_{K-1}u_{K-2}\cdots u_1}$ is already updated, $\xi^{(\ell)}_{u_{K-1}u_{K-2}\cdots u_1}$ can be used instead of $\xi^{(\ell-1)}_{u_{K-1}u_{K-2}\cdots u_1}$. Hence, (14) can be modified as

$$\xi^{(\ell)}_{u_{K-2}u_{K-3}\cdots u_0} := D^{w(0u_{K-2}u_{K-3}\cdots u_1)}\xi^{(\ell)}_{0u_{K-2}u_{K-3}\cdots u_1} + D^{w(1u_{K-2}u_{K-3}\cdots u_1)}\xi^{(\ell-1)}_{1u_{K-2}u_{K-3}\cdots u_1}. \quad (15)$$

III. ALGORITHM TO CALCULATE $\left.\frac{\partial T(D,I)}{\partial I}\right|_{I=1}$

The differential operation $\left.\frac{\partial T(D,I)}{\partial I}\right|_{I=1}$ is usually approximated by the difference operation (3). However, we will show in this section that $\left.\frac{\partial T(D,I)}{\partial I}\right|_{I=1}$ can be calculated without any difference operation.

Consider a part of a state diagram shown in Fig. 3. Let $T_a(D,I)$ and $T_b(D,I)$ be the transfer functions from the start node to nodes a and b, respectively. We define ξ_a, ξ_b, ζ_a, and ζ_b as

$$\xi_a \triangleq T_a(D,I)|_{I=1}, \quad (16)$$

$$\xi_b \triangleq T_b(D,I)|_{I=1}, \quad (17)$$

$$\zeta_a \triangleq \left.\frac{\partial T_a(D,I)}{\partial I}\right|_{I=1}, \quad (18)$$

$$\zeta_b \triangleq \left.\frac{\partial T_b(D,I)}{\partial I}\right|_{I=1}. \quad (19)$$

Then, ξ_a and ξ_b are related by

$$\xi_b = D^S\xi_a, \quad (20)$$

which corresponds to the operation of (14). On the other hand, ζ_a and ζ_b are related as follows.

$$\begin{aligned}\zeta_b &= \left.\frac{\partial T_b(D,I)}{\partial I}\right|_{I=1} \\ &= \left.\frac{\partial (D^S I^N T_a(D,I))}{\partial I}\right|_{I=1} \\ &= D^S \left. I^N \frac{\partial T_b(D,I)}{\partial I}\right|_{I=1} + N D^S T_a(D,I)|_{I=1} \\ &= D^S \zeta_a + N D^S \xi_a. \end{aligned} \tag{21}$$

The first term of (21) is the same formula as (20), which means that ζ can be calculated in the same way as ξ. The second term represents an additive term depending on N. In the case of rate $1/b$ convolutional codes, N is determined by the new coming bit u_0 as follows.

$$N = \begin{cases} 0, & \text{if } u_0 = 0 \\ 1, & \text{if } u_0 = 1. \end{cases} \tag{22}$$

Hence, $\left.\frac{\partial T(D,I)}{\partial I}\right|_{I=1}$ can be obtained by the following algorithm.

Element-wise Power Method for $T(D)|_{I=1}$ and $\left.\frac{\partial T(D,I)}{\partial I}\right|_{I=1}$

1. (*Initialization*)
   $\xi^{(0)}_{0\cdots 0} := 1$ and let other $\xi^{(0)}_{u_{K-1}u_{K-2}\cdots u_1}$'s be zero.
   Let all $\zeta^{(0)}_{u_{K-1}u_{K-2}\cdots u_1}$'s be zero.
   $\ell := 0$.

2. **repeat**

   (a) $\ell := \ell + 1$.

   (b) $\xi^{(\ell)}_{0\cdots 0} := \xi^{(\ell-1)}_{0\cdots 0}$, $\zeta^{(\ell)}_{0\cdots 0} := \zeta^{(\ell-1)}_{0\cdots 0}$.

   (c) **for** $i := 1$ **to** $2^{K-1} - 1$ **do**

   i. $u_{K-2}u_{K-3}\cdots u_0 := (i)_2^{K-1}$.

   ii.
   $$\xi^{(\ell)}_{u_{K-2}u_{K-3}\cdots u_0} := D^{w(0u_{K-2}u_{K-3}\cdots u_1)}\xi^{(\ell-1)}_{0u_{K-2}u_{K-3}\cdots u_1} + D^{w(1u_{K-2}u_{K-3}\cdots u_1)}\xi^{(\ell-1)}_{1u_{K-2}u_{K-3}\cdots u_1}. \tag{23}$$

   iii.
   $$\begin{aligned}\zeta^{(\ell)}_{u_{K-2}u_{K-3}\cdots u_0} &:= D^{w(0u_{K-2}u_{K-3}\cdots u_1)}\zeta^{(\ell-1)}_{0u_{K-2}u_{K-3}\cdots u_1} + D^{w(1u_{K-2}u_{K-3}\cdots u_1)}\zeta^{(\ell-1)}_{1u_{K-2}u_{K-3}\cdots u_1} \\ &\quad + u_0[D^{w(0u_{K-2}u_{K-3}\cdots u_1)}\xi^{(\ell-1)}_{0u_{K-2}u_{K-3}\cdots u_1} + D^{w(1u_{K-2}u_{K-3}\cdots u_1)}\xi^{(\ell-1)}_{1u_{K-2}u_{K-3}\cdots u_1}]. \end{aligned} \tag{24}$$

   until sufficient accuracy is achieved.

3.
$$\begin{aligned} T(D) &:= D^{w(100\cdots 0)}\xi_{100\cdots 0}, \\ \left.\frac{\partial T(D,I)}{\partial I}\right|_{I=1} &:= D^{w(100\cdots 0)}\zeta_{100\cdots 0}. \end{aligned}$$

which corresponds to the operation of (14). On the other hand, ζ_a and ζ_b are related as follows.

$$\begin{aligned}\zeta_b &= \left.\frac{\partial T_b(D,I)}{\partial I}\right|_{I=1} \\ &= \left.\frac{\partial (D^S I^N T_a(D,I))}{\partial I}\right|_{I=1} \\ &= D^S\, I^N \left.\frac{\partial T_b(D,I)}{\partial I}\right|_{I=1} + N D^S T_a(D,I)|_{I=1} \\ &= D^S \zeta_a + N D^S \xi_a. \end{aligned} \tag{21}$$

The first term of (21) is the same formula as (20), which means that ζ can be calculated in the same way as ξ. The second term represents an additive term depending on N. In the case of rate $1/b$ convolutional codes, N is determined by the new coming bit u_0 as follows.

$$N = \begin{cases} 0, & \text{if } u_0 = 0 \\ 1, & \text{if } u_0 = 1. \end{cases} \tag{22}$$

Hence, $\left.\frac{\partial T(D,I)}{\partial I}\right|_{I=1}$ can be obtained by the following algorithm.

Element-wise Power Method for $T(D)|_{I=1}$ and $\left.\frac{\partial T(D,I)}{\partial I}\right|_{I=1}$

1. (*Initialization*)
   $\xi^{(0)}_{0\cdots 0} := 1$ and let other $\xi^{(0)}_{u_{K-1}u_{K-2}\cdots u_1}$'s be zero.
   Let all $\zeta^{(0)}_{u_{K-1}u_{K-2}\cdots u_1}$'s be zero.
   $\ell := 0$.

2. **repeat**

   (a) $\ell := \ell + 1$.

   (b) $\xi^{(\ell)}_{0\cdots 0} := \xi^{(\ell-1)}_{0\cdots 0}$, $\zeta^{(\ell)}_{0\cdots 0} := \zeta^{(\ell-1)}_{0\cdots 0}$.

   (c) **for** $i := 1$ **to** $2^{K-1} - 1$ **do**

   i. $u_{K-2}u_{K-3}\cdots u_0 := (i)_2^{K-1}$.

   ii.
   $$\xi^{(\ell)}_{u_{K-2}u_{K-3}\cdots u_0} := D^{w(0u_{K-2}u_{K-3}\cdots u_1)}\xi^{(\ell-1)}_{0u_{K-2}u_{K-3}\cdots u_1} + D^{w(1u_{K-2}u_{K-3}\cdots u_1)}\xi^{(\ell-1)}_{1u_{K-2}u_{K-3}\cdots u_1}. \tag{23}$$

   iii.
   $$\begin{aligned}\zeta^{(\ell)}_{u_{K-2}u_{K-3}\cdots u_0} &:= D^{w(0u_{K-2}u_{K-3}\cdots u_1)}\zeta^{(\ell-1)}_{0u_{K-2}u_{K-3}\cdots u_1} + D^{w(1u_{K-2}u_{K-3}\cdots u_1)}\zeta^{(\ell-1)}_{1u_{K-2}u_{K-3}\cdots u_1} \\ &\quad + u_0[D^{w(0u_{K-2}u_{K-3}\cdots u_1)}\xi^{(\ell-1)}_{0u_{K-2}u_{K-3}\cdots u_1} + D^{w(1u_{K-2}u_{K-3}\cdots u_1)}\xi^{(\ell-1)}_{1u_{K-2}u_{K-3}\cdots u_1}]. \end{aligned} \tag{24}$$

   until sufficient accuracy is achieved.

3.
$$\begin{aligned} T(D) &:= D^{w(100\cdots 0)}\xi_{100\cdots 0}, \\ \left.\frac{\partial T(D,I)}{\partial I}\right|_{I=1} &:= D^{w(100\cdots 0)}\zeta_{100\cdots 0}. \end{aligned}$$

In order to improve the convergence speed, we can use (15) instead of (23). Furthermore, instead of (24), we can use

$$\zeta^{(\ell)}_{u_{K-2}u_{K-3}\cdots u_0} := D^{w(0u_{K-2}u_{K-3}\cdots u_1)}\zeta^{(\ell)}_{0u_{K-2}u_{K-3}\cdots u_1} + D^{w(1u_{K-2}u_{K-3}\cdots u_1)}\zeta^{(\ell-1)}_{1u_{K-2}u_{K-3}\cdots u_1}$$
$$+u_0[D^{w(0u_{K-2}u_{K-3}\cdots u_1)}\xi^{(\ell)}_{0u_{K-2}u_{K-3}\cdots u_1} + D^{w(1u_{K-2}u_{K-3}\cdots u_1)}\xi^{(\ell-1)}_{1u_{K-2}u_{K-3}\cdots u_1}]. \quad (25)$$

III. CONCLUDING REMARKS

In this paper, we established the numerical calculation algorithm for the transfer function bounds of convolutional codes. This algorithm was originally found and used to calculate theoretical bounds of error probability in [5] though it has not been published in any paper or report. For simplicity, the algorithm was explained for rate $1/n$ convolutional codes in the foregoing sections. However, the algorithm can easily be extended for any b/n convolutional codes. Furthermore, the algorithm can be applied for the transfer function bounds of TCM schemes. If a TCM scheme is uniform, then the algorithm can be applied straightforwardly. For a nonuniform TCM scheme, the labels of state diagrams become matrices. Hence, by extending ξ and ζ of each node to matrices, the algorithm can also be used for nonuniform TCM schemes.

ACKNOWLEDGMENT

The authors thank Prof. Ikuo Oka and Prof. Tsutomu Kawabata for discussion.

References

[1] A. J. Viterbi, "Principles of Digital Communication and Coding", *McGraw-Hill Book Company*, 1979

[2] E. Biglieri, D. Divsalar, P. J. MacLane, and M. K. Simon, "Introduction to Trellis–Coded Modulation with Applications", *Macmillan Publishing Company*, 1991

[3] I. Oka and E. Biglieri, "Error Matrix and Pairwise–State Methods in the Error Probability Analysis of Trellis Coded Modulation", *IEICE Trans.*, vol.J73-B-I, no.12, pp.939–942, Dec. 1990 (in Japanese)

[4] R. L. Burden, J. D. Faires, and A. C. Reynolds, "Numerical Analysis (second ed.)", *Pindle, Weber & Schmidt*, 1981

[5] H. Yamamoto and K. Itoh, "Viterbi Decoding Algorithm for Convolutional Codes with Repeat Request", *IEEE Trans. on Inform. Theory*, vol.IT–26, no.5, pp.540–547, Sep. 1980

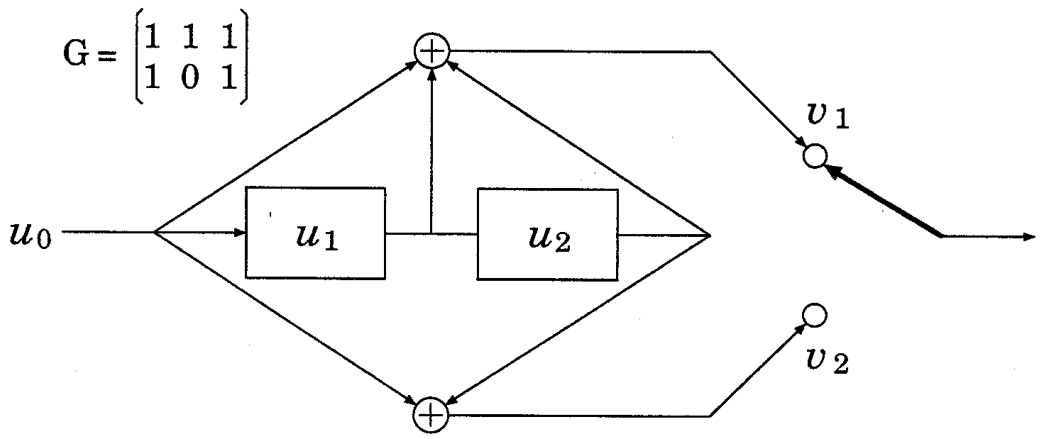

Figure 1: Encoder of a convolutional code.

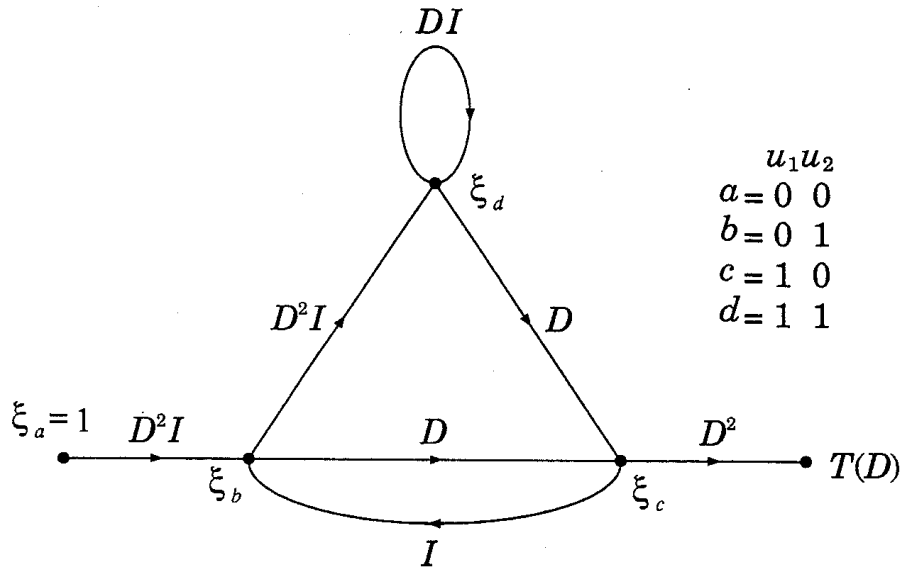

Figure 2: State diagram.

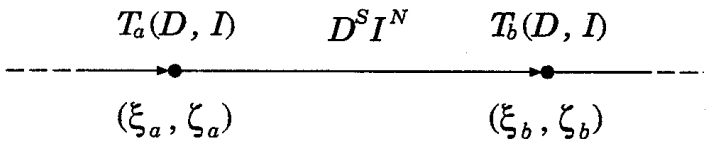

Figure 3: A part of state diagram.