CHAPTER 11

Convolutional Codes

As discussed in Chapter 1, convolutional codes differ from block codes in that the encoder contains memory, and the encoder outputs at any given time unit depend not only on the inputs at that time unit but also on some number of previous inputs. A rate $R = k/n$ convolutional encoder with memory order m can be realized as a k-input, n-output linear sequential circuit with input memory m; that is, inputs remain in the encoder for an additional m time units after entering. Typically, n and k are small integers, $k < n$, the information sequence is divided into blocks of length k, and the codeword is divided into blocks of length n. In the important special case when $k = 1$, the information sequence is not divided into blocks and is processed continuously. Unlike with block codes, large minimum distances and low error probabilities are achieved not by increasing k and n but by increasing the memory order m.

In this chapter we describe the convolutional encoding process and explain the notation used to represent convolutional codes. We show how convolutional encoders can be represented using a state diagram and how weight-enumerating functions for convolutional codes can be derived. Finally, we introduce several distance measures for convolutional codes.

Convolutional codes were first introduced by Elias [1] in 1955 as an alternative to block codes. Shortly thereafter, Wozencraft and Reiffen [2] proposed sequential decoding as an efficient decoding method for convolutional codes with large constraint lengths, and experimental studies soon began to appear. In 1963 Massey [3] proposed a less efficient but simpler-to-implement decoding method called threshold decoding. This advance spawned a number of practical applications of convolutional codes to digital transmission over telephone, satellite, and radio channels. These two suboptimal decoding methods are discussed in detail in Chapter 13. Then, in 1967 Viterbi [4] proposed a maximum likelihood (ML) decoding algorithm that was relatively easy to implement for soft-decision decoding of convolutional codes with small constraint lengths. The Viterbi algorithm, along with soft-decision versions of sequential decoding, led to the application of convolutional codes to deep-space and satellite communication systems in the 1970s. In 1974, Bahl, Cocke, Jelinek, and Raviv (BCJR) [5] introduced a maximum a posteriori probability (MAP) decoding algorithm for convolutional codes with unequal a priori probabilities for the information bits. The BCJR algorithm has been applied in recent years to soft-decision iterative decoding schemes in which the a priori probabilities of the information bits change from iteration to iteration. These two optimal decoding algorithms are discussed in detail in Chapter 12, and the associated iterative decoding schemes are presented in Chapter 16.

In 1976 Ungerboeck and Csajka [6] introduced the concept of trellis-coded modulation using short constraint length convolutional codes of rate $R = k/(k+1)$ along with soft-decision Viterbi decoding. (For a more complete introduction, see

Ungerboeck [7].) These schemes form the basis for achieving high-speed digital transmission over bandwidth-limited telephone channels and are discussed in detail in Chapter 18. Ungerboeck's work spawned interest in the properties of feedback encoders for convolutional codes, since high-rate codes are most efficiently represented in systematic feedback form. Another application of systematic feedback convolutional encoders, called *turbo coding*, was introduced by Berrou, Glavieux, and Thitimajshima [8] in 1993. (For a more complete introduction, see Berrou and Glavieux [9].) This coding scheme, which combines a parallel concatenation of two systematic feedback encoders with iterative MAP decoding, is capable of achieving moderately low BERs of around 10^{-5} at SNRs near the Shannon limit. It is currently being considered for adoption in a variety of applications, including data transmission on digital cellular channels, deep-space and satellite communication, and high-speed digital magnetic recording. Turbo coding is discussed in detail in Chapter 16.

Convolutional codes are covered in varying amounts of detail in many books on coding theory and digital communications. Recently, three books have appeared devoted solely to convolutional codes [10, 11, 12]. The book by Johannesson and Zigangirov [12] is the most comprehensive and contains a detailed analysis of all aspects of convolutional codes.

11.1 ENCODING OF CONVOLUTIONAL CODES

We present the basic elements of encoding for convolutional codes using a series of examples. Encoders for convolutional codes fall into two general categories: feedforward and feedback. Further, within each category, encoders can be either systematic or nonsystematic. We begin by considering the class of *nonsystematic feedforward encoders.*

EXAMPLE 11.1 A Rate $R = 1/2$ Nonsystematic Feedforward Convolutional Encoder

A block diagram of a binary rate $R = 1/2$ nonsystematic feedforward convolutional encoder with memory order $m = 3$ is shown in Figure 11.1. Note that the encoder consists of $k = 1$ shift register with $m = 3$ delay elements and with $n = 2$ modulo-2 adders. The mod-2 adders can be implemented as EXCLUSIVE-OR gates. (We note here that any multi-input mod-2 adder, such as those in Figure 11.1, can be implemented using a sequence of binary input mod-2 adders, or EXCLUSIVE-OR gates.) Because mod-2 addition is a linear operation, the encoder is a linear system. All convolutional codes can be realized using a linear feedforward shift register encoder of this type.

The *information sequence* $\mathbf{u} = (u_0, u_1, u_2, \cdots)$ enters the encoder one bit at a time. Because the encoder is a linear system, the two encoder *output sequences* $\mathbf{v}^{(0)} = (v_0^{(0)}, v_1^{(0)}, v_2^{(0)}, \cdots)$ and $\mathbf{v}^{(1)} = (v_0^{(1)}, v_1^{(1)}, v_2^{(1)}, \cdots)$ can be obtained as the convolution of the input sequence $\mathbf{u}$ with the two encoder *impulse responses*. The impulse responses are obtained by letting $\mathbf{u} = (1\ 0\ 0 \cdots)$ and observing the two output sequences. For an encoder with memory order m, the impulse responses can last at most $m + 1$ time units and are written as $\mathbf{g}^{(0)} = (g_0^{(0)}, g_1^{(0)}, \cdots, g_m^{(0)})$ and

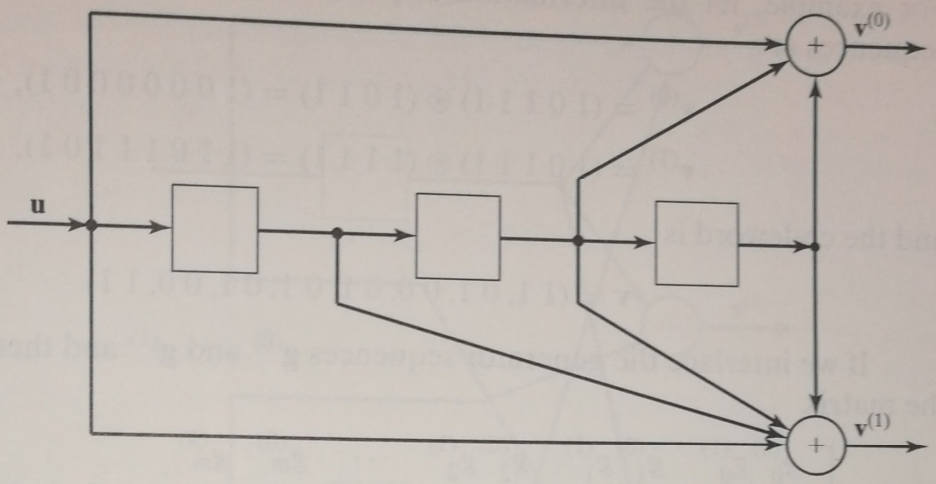


FIGURE 11.1: A rate $R = 1/2$ binary nonsystematic feedforward convolutional encoder with memory order $m = 3$.

$\mathbf{g}^{(1)} = (g_0^{(1)}, g_1^{(1)}, \cdots, g_m^{(1)})$. For the encoder of Figure 11.1,

$$\mathbf{g}^{(0)} = (1\ 0\ 1\ 1) \tag{11.1a}$$

$$\mathbf{g}^{(1)} = (1\ 1\ 1\ 1). \tag{11.1b}$$

The impulse responses $\mathbf{g}^{(0)}$ and $\mathbf{g}^{(1)}$ are called the *generator sequences* of the encoder.

We can now write the *encoding equations* as

$$\mathbf{v}^{(0)} = \mathbf{u} \circledast \mathbf{g}^{(0)}, \tag{11.2a}$$

$$\mathbf{v}^{(1)} = \mathbf{u} \circledast \mathbf{g}^{(1)}, \tag{11.2b}$$

where $\circledast$ denotes discrete convolution, and all operations are modulo-2. The convolution operation implies that for all $l \geq 0$,

$$v_l^{(j)} = \sum_{i=0}^{m} u_{l-i} g_i^{(j)} = u_l g_0^{(j)} + u_{l-1} g_1^{(j)} + \cdots + u_{l-m} g_m^{(j)}, \quad j = 0, 1, \tag{11.3}$$

where $u_{l-i} \triangleq 0$ for all $l < i$, and all operations are modulo-2. Hence, for the encoder of Figure 11.1,

$$v_l^{(0)} = u_l + u_{l-2} + u_{l-3} \tag{11.4a}$$

$$v_l^{(1)} = u_l + u_{l-1} + u_{l-2} + u_{l-3}, \tag{11.4b}$$

as can easily be verified by direct inspection of the encoder diagram. After encoding, the two output sequences are multiplexed into a single sequence, called the *codeword*, for transmission over the channel. The codeword is given by

$$\mathbf{v} = (v_0^{(0)} v_0^{(1)}, v_1^{(0)} v_1^{(1)}, v_2^{(0)} v_2^{(1)}, \cdots). \tag{11.5}$$

For example, let the information sequence $\mathbf{u} = (1\ 0\ 1\ 1\ 1)$. Then, the output sequences are

$$\mathbf{v}^{(0)} = (1\ 0\ 1\ 1\ 1) \circledast (1\ 0\ 1\ 1) = (1\ 0\ 0\ 0\ 0\ 0\ 0\ 1), \tag{11.6a}$$

$$\mathbf{v}^{(1)} = (1\ 0\ 1\ 1\ 1) \circledast (1\ 1\ 1\ 1) = (1\ 1\ 0\ 1\ 1\ 1\ 0\ 1), \tag{11.6b}$$

and the codeword is

$$\mathbf{v} = (1\ 1, 0\ 1, 0\ 0, 0\ 1, 0\ 1, 0\ 1, 0\ 0, 1\ 1). \tag{11.7}$$

If we interlace the generator sequences $\mathbf{g}^{(0)}$ and $\mathbf{g}^{(1)}$ and then arrange them in the matrix

$$\mathbf{G} = \begin{bmatrix} g_0^{(0)} g_0^{(1)} & g_1^{(0)} g_1^{(1)} & g_2^{(0)} g_2^{(1)} & \cdots & g_m^{(0)} & g_m^{(1)} & & \\ & g_0^{(0)} g_0^{(1)} & g_1^{(0)} g_1^{(1)} & \cdots & g_{m-1}^{(0)} g_{m-1}^{(1)} & g_m^{(0)} & g_m^{(1)} & \\ & & g_0^{(0)} g_0^{(1)} & \cdots & g_{m-2}^{(0)} g_{m-2}^{(1)} & g_{m-1}^{(0)} g_{m-1}^{(1)} & g_m^{(0)} g_m^{(1)} & \\ & & & \ddots & & & \ddots & \end{bmatrix}, \tag{11.8}$$

where the blank areas are all zeros, we can rewrite the encoding equations in matrix form as

$$\mathbf{v} = \mathbf{u}\mathbf{G}, \tag{11.9}$$

where again all operations are modulo-2. The matrix $\mathbf{G}$ is called the (time domain) *generator matrix* of the encoder. Note that each row of $\mathbf{G}$ is identical to the previous row but shifted $n = 2$ places to the right and that $\mathbf{G}$ is a semi-infinite matrix, corresponding to the fact that the information sequence $\mathbf{u}$ is of arbitrary length. If $\mathbf{u}$ has finite length h, then $\mathbf{G}$ has h rows and $2(m + h)$ columns, and $\mathbf{v}$ has length $2(m + h)$. For the encoder of Figure 11.1, if $\mathbf{u} = (1\ 0\ 1\ 1\ 1)$, then

$$\begin{aligned} \mathbf{v} &= \mathbf{u}\mathbf{G} \\ &= (1\ 0\ 1\ 1\ 1) \begin{bmatrix} 1\,1 & 0\,1 & 1\,1 & 1\,1 & & & & \\ & 1\,1 & 0\,1 & 1\,1 & 1\,1 & & & \\ & & 1\,1 & 0\,1 & 1\,1 & 1\,1 & & \\ & & & 1\,1 & 0\,1 & 1\,1 & 1\,1 & \\ & & & & 1\,1 & 0\,1 & 1\,1 & 1\,1 \end{bmatrix} \\ &= (1\ 1, 0\ 1, 0\ 0, 0\ 1, 0\ 1, 0\ 1, 0\ 0, 1\ 1), \end{aligned} \tag{11.10}$$

which agrees with our previous calculation using discrete convolution.

EXAMPLE 11.2 A Rate $R = 2/3$ Nonsystematic Feedforward Convolutional Encoder

As a second example of a nonsystematic feedforward convolutional encoder, consider the binary rate $R = 2/3$ encoder with memory order $m = 1$ shown in Figure 11.2. The encoder consists of $k = 2$ shift registers, each with $m = 1$

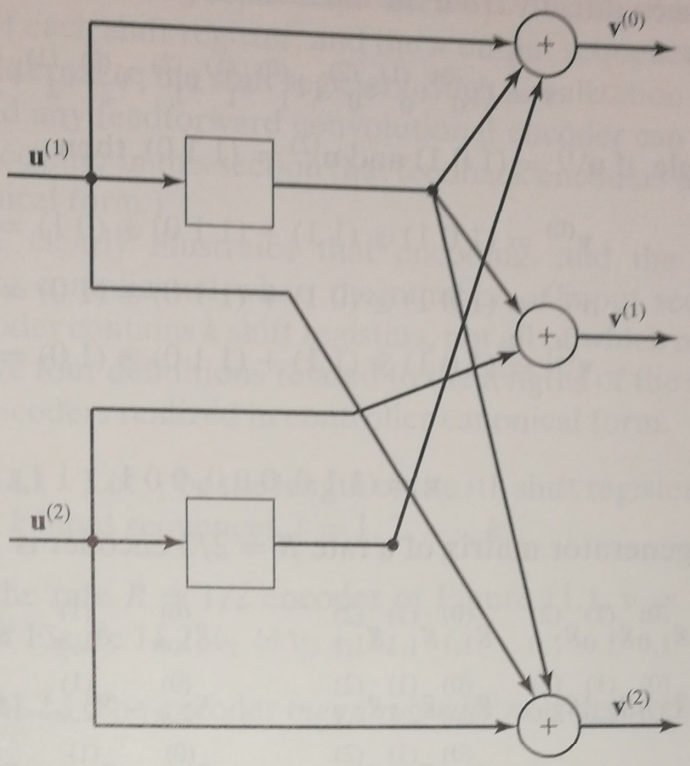


FIGURE 11.2: A rate $R = 2/3$ binary nonsystematic feedforward convolutional encoder with memory order $m = 1$.

delay element, along with $n = 3$ mod-2 adders. The information sequence enters the encoder $k = 2$ bits at a time and can be written as $\mathbf{u} = (\mathbf{u}_0, \mathbf{u}_1, \cdots) = (u_0^{(1)}u_0^{(2)}, u_1^{(1)}u_1^{(2)}, u_2^{(1)}u_2^{(2)}, \cdots)$ or as the two input sequences $\mathbf{u}^{(1)} = (u_0^{(1)}, u_1^{(1)}, u_2^{(1)}, \cdots)$ and $\mathbf{u}^{(2)} = (u_0^{(2)}, u_1^{(2)}, u_2^{(2)}, \cdots)$. There are three generator sequences corresponding to each input sequence. Letting $\mathbf{g}_i^{(j)} = (g_{i,0}^{(j)}, g_{i,1}^{(j)}, \cdots, g_{i,m}^{(j)})$ represent the generator sequence corresponding to input i and output j, we obtain the generator sequences for the encoder of Figure 11.2;

$$\mathbf{g}_1^{(0)} = (1\ 1) \qquad \mathbf{g}_1^{(1)} = (0\ 1) \qquad \mathbf{g}_1^{(2)} = (1\ 1), \tag{11.11a}$$

$$\mathbf{g}_2^{(0)} = (0\ 1) \qquad \mathbf{g}_2^{(1)} = (1\ 0) \qquad \mathbf{g}_2^{(2)} = (1\ 0), \tag{11.11b}$$

and we can write the encoding equations as

$$\mathbf{v}^{(0)} = \mathbf{u}^{(1)} \circledast \mathbf{g}_1^{(0)} + \mathbf{u}^{(2)} \circledast \mathbf{g}_2^{(0)} \tag{11.12a}$$

$$\mathbf{v}^{(1)} = \mathbf{u}^{(1)} \circledast \mathbf{g}_1^{(1)} + \mathbf{u}^{(2)} \circledast \mathbf{g}_2^{(1)} \tag{11.12b}$$

$$\mathbf{v}^{(2)} = \mathbf{u}^{(1)} \circledast \mathbf{g}_1^{(2)} + \mathbf{u}^{(2)} \circledast \mathbf{g}_2^{(2)}. \tag{11.12c}$$

The convolution operation implies that

$$v_l^{(0)} = u_l^{(1)} + \quad + u_{l-1}^{(1)} + u_{l-1}^{(2)}, \tag{11.13a}$$

$$v_l^{(1)} = \quad u_l^{(2)} + u_{l-1}^{(1)}, \tag{11.13b}$$

$$v_l^{(2)} = u_l^{(1)} + u_l^{(2)} + u_{l-1}^{(1)}, \tag{11.13c}$$

as can be seen directly from the encoder diagram. After multiplexing, the codeword is given by

$$\mathbf{v} = (v_0^{(0)} v_0^{(1)} v_0^{(2)}, v_1^{(0)} v_1^{(1)} v_1^{(2)}, v_2^{(0)} v_2^{(1)} v_2^{(2)}, \cdots). \tag{11.14}$$

For example, if $\mathbf{u}^{(1)} = (1\ 0\ 1)$ and $\mathbf{u}^{(2)} = (1\ 1\ 0)$, then

$$\mathbf{v}^{(0)} = (1\ 0\ 1) \circledast (1\ 1) + (1\ 1\ 0) \circledast (0\ 1) = (1\ 0\ 0\ 1), \tag{11.15a}$$

$$\mathbf{v}^{(1)} = (1\ 0\ 1) \circledast (0\ 1) + (1\ 1\ 0) \circledast (1\ 0) = (1\ 0\ 0\ 1), \tag{11.15b}$$

$$\mathbf{v}^{(2)} = (1\ 0\ 1) \circledast (1\ 1) + (1\ 1\ 0) \circledast (1\ 0) = (0\ 0\ 1\ 1), \tag{11.15c}$$

and

$$\mathbf{v} = (1\ 1\ 0, 0\ 0\ 0, 0\ 0\ 1, 1\ 1\ 1). \tag{11.16}$$

The generator matrix of a rate $R = 2/3$ encoder is

$$\mathbf{G} = \begin{bmatrix} g_{1,0}^{(0)} g_{1,0}^{(1)} g_{1,0}^{(2)} & g_{1,1}^{(0)} g_{1,1}^{(1)} g_{1,1}^{(2)} & \cdots & g_{1,m}^{(0)} & g_{1,m}^{(1)} & g_{1,m}^{(2)} & & \\ g_{2,0}^{(0)} g_{2,0}^{(1)} g_{2,0}^{(2)} & g_{2,1}^{(0)} g_{2,1}^{(1)} g_{2,1}^{(2)} & \cdots & g_{2,m}^{(0)} & g_{2,m}^{(1)} & g_{2,m}^{(2)} & & \\ & g_{1,0}^{(0)} g_{1,0}^{(1)} g_{1,0}^{(2)} & \cdots & g_{1,m-1}^{(0)} & g_{1,m-1}^{(1)} & g_{1,m-1}^{(2)} & g_{1,m}^{(0)} g_{1,m}^{(1)} g_{1,m}^{(2)} & \\ & g_{2,0}^{(0)} g_{2,0}^{(1)} g_{2,0}^{(2)} & \cdots & g_{2,m-1}^{(0)} & g_{2,m-1}^{(1)} & g_{2,m-1}^{(2)} & g_{2,m}^{(0)} g_{2,m}^{(1)} g_{2,m}^{(2)} & \\ & \ddots & & & & & \ddots & \end{bmatrix}, \tag{11.17}$$

and the encoding equations in matrix form are again given by $\mathbf{v} = \mathbf{uG}$. Note that each set of $k = 2$ rows of $\mathbf{G}$ is identical to the previous set of rows but shifted $n = 3$ places to the right. For the encoder of Figure 11.2, if $\mathbf{u}^{(1)} = (1\ 0\ 1)$ and $\mathbf{u}^{(2)} = (1\ 1\ 0)$, then $\mathbf{u} = (\mathbf{u}_0, \mathbf{u}_1, \mathbf{u}_2) = (1\ 1, 0\ 1, 1\ 0)$, and

$$\mathbf{v} = \mathbf{uG} = (1\ 1, 0\ 1, 1\ 0) \begin{bmatrix} 1\ 0\ 1 & 1\ 1\ 1 & & \\ 0\ 1\ 1 & 1\ 0\ 0 & & \\ & 1\ 0\ 1 & 1\ 1\ 1 & \\ & 0\ 1\ 1 & 1\ 0\ 0 & \\ & & 1\ 0\ 1 & 1\ 1\ 1 \\ & & 0\ 1\ 1 & 1\ 0\ 0 \end{bmatrix} = (1\ 1\ 0, 0\ 0\ 0, 0\ 0\ 1, 1\ 1\ 1), \tag{11.18}$$

which again agrees with our previous calculation using discrete convolution.

In Figures 11.1 and 11.2, the connections from the k shift registers used to store the input sequences to the n modulo-2 adders used to form the output sequences correspond directly to the nonzero terms in the kn generator sequences given in (11.1) and (11.11), respectively. Similarly, any set of kn generator sequences can be used to realize a rate $R = k/n$ feedforward convolutional encoder of the same type shown in Figures 11.1 and 11.2 with k shift registers to store the input sequences and

n modulo-2 adders to form the output sequences. Specifically, the k input sequences enter the left end of each shift register, and the n output sequences are produced by modulo-2 adders external to the shift registers. Such a realization is called *controller canonical form*, and any feedforward convolutional encoder can be realized in this manner. (We will see later in this section that feedback encoders also can be realized in controller canonical form.)

Example 11.2 clearly illustrates that encoding, and the notation used to describe it, is more complicated when the number of input sequences $k > 1$. In particular, the encoder contains k shift registers, not all of which must have the same length. We now give four definitions related to the lengths of the shift registers used in convolutional encoders realized in controller canonical form.

DEFINITION 11.1 Let ν_i be the length of the ith shift register in a convolutional encoder with k input sequences, $i = 1, 2, \cdots, k$.

For example, for the rate $R = 1/2$ encoder of Figure 11.1, $\nu = 3$, and for the rate $R = 2/3$ encoder of Figure 11.2, $\nu_1 = \nu_2 = 1$.

DEFINITION 11.2 The encoder *memory order* m is defined as

$$m = \max_{1 \le i \le k} \nu_i; \tag{11.19}$$

that is, m is the maximum length of all k shift registers.

For the rate $R = 1/2$ encoder of Figure 11.1, $m = \nu_1 = 3$, and for the rate $R = 2/3$ encoder of Figure 11.2, $m = \max(\nu_1, \nu_2) = \max(1, 1) = 1$.

DEFINITION 11.3 The *overall constraint length* ν of the encoder is defined as

$$\nu = \sum_{1 \le i \le k} \nu_i; \tag{11.20}$$

that is, ν is the sum of the lengths of all k shift registers.

For the rate $R = 1/2$ encoder of Figure 11.1, $\nu = \nu_1 = 3$, and for the rate $R = 2/3$ encoder of Figure 11.2, $\nu = \nu_1 + \nu_2 = 1 + 1 = 2$.

DEFINITION 11.4 A rate $R = k/n$ convolutional encoder with overall constraint length ν is referred to as an (n, k, ν) encoder.

The rate $R = 1/2$ encoder of Figure 11.1 is a $(2, 1, 3)$, encoder and the rate $R = 2/3$ encoder of Figure 11.2 is a $(3, 2, 2)$ encoder.

Here we note that for an $(n, 1, \nu)$ encoder, the overall constraint length ν equals the memory order m; that is, $\nu = m$ when $k = 1$; however, for $k > 1$, in general, $m \le \nu$, and in the case when each shift register has the same length, $\nu = km$.

EXAMPLE 11.3 A (4, 3, 3) Nonsystematic Feedforward Convolutional Encoder

A $(4, 3, 3)$ binary nonsystematic feedforward convolutional encoder in which the shift register lengths are $\nu_1 = 0$, $\nu_2 = 1$, and $\nu_3 = 2$ is shown in Figure 11.3. The

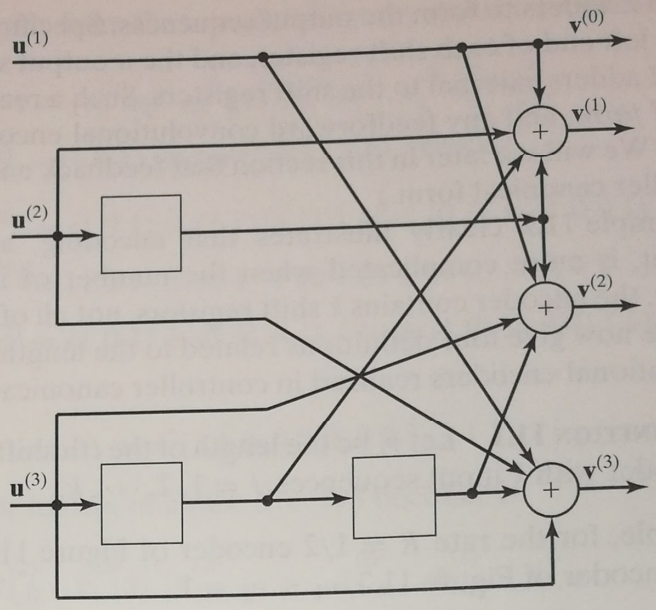


FIGURE 11.3: A (4, 3, 3) binary nonsystematic feedforward convolutional encoder.

memory order is $m = 2$, and the overall constraint length is $\nu = 3$. Following the notation of Example 11.2, the generator sequences are given by

$$\mathbf{g}_1^{(0)} = (100) \quad \mathbf{g}_1^{(1)} = (100) \quad \mathbf{g}_1^{(2)} = (100) \quad \mathbf{g}_1^{(3)} = (100), \tag{11.21a}$$

$$\mathbf{g}_2^{(0)} = (000) \quad \mathbf{g}_2^{(1)} = (110) \quad \mathbf{g}_2^{(2)} = (010) \quad \mathbf{g}_2^{(3)} = (100), \tag{11.21b}$$

$$\mathbf{g}_3^{(0)} = (000) \quad \mathbf{g}_3^{(1)} = (010) \quad \mathbf{g}_3^{(2)} = (101) \quad \mathbf{g}_3^{(3)} = (101). \tag{11.21c}$$

In the general case of an (n, k, ν) feedforward encoder with memory order m, the generator matrix is given by

$$\mathbf{G} = \begin{bmatrix} \mathbf{G}_0 & \mathbf{G}_1 & \mathbf{G}_2 & \cdots & \mathbf{G}_m & & \\ & \mathbf{G}_0 & \mathbf{G}_1 & \cdots & \mathbf{G}_{m-1} & \mathbf{G}_m & \\ & & \mathbf{G}_0 & \cdots & \mathbf{G}_{m-2} & \mathbf{G}_{m-1} & \mathbf{G}_m \\ & & \ddots & & & & \end{bmatrix}, \tag{11.22}$$

where each $\mathbf{G}_l$ is a $k \times n$ submatrix whose entries are

$$\mathbf{G}_l = \begin{bmatrix} g_{1,l}^{(0)} & g_{1,l}^{(1)} & \cdots & g_{1,l}^{(n-1)} \\ g_{2,l}^{(0)} & g_{2,l}^{(1)} & \cdots & g_{2,l}^{(n-1)} \\ \vdots & \vdots & & \vdots \\ g_{k,l}^{(0)} & g_{k,l}^{(1)} & \cdots & g_{k,l}^{(n-1)} \end{bmatrix}. \tag{11.23}$$

Again note that each set of k rows of $\mathbf{G}$ is identical to the previous set of rows but shifted n places to the right. For an information sequence $\mathbf{u} = (\mathbf{u}_0, \mathbf{u}_1, \cdots) = (u_0^{(1)} u_0^{(2)} \cdots u_0^{(k)}, u_1^{(1)} u_1^{(2)} \cdots u_1^{(k)}, \cdots)$, the codeword

$\mathbf{v} = (\mathbf{v}_0, \mathbf{v}_1, \cdots) = (v_0^{(0)} v_0^{(1)} \cdots v_0^{(n-1)}, v_1^{(0)} v_1^{(1)} \cdots v_1^{(n-1)}, \cdots)$ is given by $\mathbf{v} = \mathbf{uG}$. We are now in a position to give a formal definition of a convolutional code.

DEFINITION 11.5 An (n, k, ν) convolutional code is the set of all output sequences (codewords) produced by an (n, k, ν) convolutional encoder; that is, it is the row space of the encoder generator matrix $\mathbf{G}$.

Because the codeword $\mathbf{v}$ is a linear combination of rows of the generator matrix $\mathbf{G}$, an (n, k, ν) convolutional code is a linear code.

EXAMPLE 11.3 (Continued)

The set of all codewords produced by the $(4, 3, 3)$ convolutional encoder of Figure 11.3 is a $(4, 3, 3)$ convolutional code.

In any linear system, time-domain operations involving convolution can be replaced by more convenient transform-domain operations involving polynomial multiplication. Because a convolutional encoder is a linear system, each sequence in the encoding equations can be replaced by a corresponding polynomial, and the convolution operation replaced by polynomial multiplication. In the polynomial representation of a binary sequence, the sequence itself is represented by the coefficients of the polynomial. For example, for a $(2, 1, \nu)$ encoder, the encoding equations become

$$\mathbf{v}^{(0)}(D) = \mathbf{u}(D)\mathbf{g}^{(0)}(D), \tag{11.24a}$$

$$\mathbf{v}^{(1)}(D) = \mathbf{u}(D)\mathbf{g}^{(1)}(D), \tag{11.24b}$$

where

$$\mathbf{u}(D) = u_0 + u_1 D + u_2 D^2 + \cdots \tag{11.25a}$$

is the information sequence,

$$\mathbf{v}^{(0)}(D) = v_0^{(0)} + v_1^{(0)} D + v_2^{(0)} D^2 + \cdots \tag{11.25b}$$

and

$$\mathbf{v}^{(1)}(D) = v_0^{(1)} + v_1^{(1)} D + v_2^{(1)} D^2 + \cdots \tag{11.25c}$$

are the encoded sequences,

$$\mathbf{g}^{(0)}(D) = g_0^{(0)} + g_1^{(0)} D + \cdots + g_m^{(0)} D^m \tag{11.25d}$$

and

$$\mathbf{g}^{(1)}(D) = g_0^{(1)} + g_1^{(1)} D + \cdots + g_m^{(1)} D^m \tag{11.25e}$$

are the *generator polynomials* of the code, and all operations are modulo-2. We can then write the codeword as

$$\mathbf{V}(D) = [\mathbf{v}^0(D), \mathbf{v}^1(D)] \tag{11.26a}$$

or, after multiplexing, as

$$\mathbf{v}(D) = \mathbf{v}^{(0)}(D^2) + D\mathbf{v}^{(1)}(D^2). \tag{11.26b}$$

The indeterminate D can be interpreted as a delay operator, where the power of D denotes the number of time units a bit is delayed with respect to the initial bit in the sequence.

EXAMPLE 11.1 (Continued)

For the $(2, 1, 3)$ encoder of Figure 11.1, the generator polynomials are $\mathbf{g}^{(0)}(D) = 1 + D^2 + D^3$ and $\mathbf{g}^{(1)}(D) = 1 + D + D^2 + D^3$. For the information sequence $\mathbf{u}(D) = 1 + D^2 + D^3 + D^4$, the encoding equations are

$$\mathbf{v}^{(0)}(D) = (1 + D^2 + D^3 + D^4)(1 + D^2 + D^3) = 1 + D^7, \tag{11.27a}$$

$$\mathbf{v}^{(1)}(D) = (1 + D^2 + D^3 + D^4)(1 + D + D^2 + D^3) = 1 + D + D^3 + D^4 + D^5 + D^7, \tag{11.27b}$$

and we can write the codeword as

$$\mathbf{V}(D) = [1 + D^7, 1 + D + D^3 + D^4 + D^5 + D^7], \tag{11.28a}$$

or

$$\mathbf{v}(D) = 1 + D + D^3 + D^7 + D^9 + D^{11} + D^{14} + D^{15}. \tag{11.28b}$$

Note that in each case the result is the same as previously computed using convolution and matrix multiplication.

The generator polynomials of an encoder realized in controller canonical form can be determined directly from the encoder block diagram, as noted previously for generator sequences. The sequence of connections (a 1 representing a connection and a 0 no connection) from the delay elements of a shift register to an output (mod-2 adder) is the sequence of coefficients in the associated generator polynomial; that is, it is the generator sequence. The lowest-degree (constant) terms in the generator polynomials correspond to the connections at the left ends of the shift registers, whereas the highest-degree terms correspond to the connections at the right ends of the shift registers. For example, in Figure 11.1, the sequence of connections from the shift register to output $\mathbf{v}^{(0)}$ is $\mathbf{g}^{(0)} = (1\ 0\ 1\ 1)$, and the corresponding generator polynomial is $\mathbf{g}^{(0)}(D) = 1 + D^2 + D^3$.

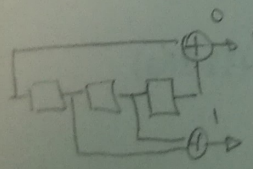
Because the right end of the shift register in an $(n, 1, \nu)$ encoder realized in controller canonical form must be connected to at least one output, the degree of at least one generator polynomial must be equal to the shift register length $\nu_1 = \nu = m$; that is,

$$m = \max_{0 \le j \le n-1} [\deg\ \mathbf{g}^{(j)}(D)]. \tag{11.29}$$

In an (n, k, ν) encoder where $k > 1$, there are n generator polynomials for each of the k inputs. Each set of n generators represents the sequences of connections from one of the shift registers to the n outputs, and hence

$$\nu_i = \max_{0 \le j \le n-1} [\deg\ \mathbf{g}_i^{(j)}(D)], \quad 1 \le i \le k, \tag{11.30}$$

where $\mathbf{g}_i^{(j)}(D)$ is the generator polynomial relating the ith input to the jth output.



Because the encoder is a linear system, and $\mathbf{u}^{(i)}(D)$ represents the ith input sequence, and $\mathbf{v}^{(j)}(D)$ represents the jth output sequence, the generator polynomial $\mathbf{g}_i^{(j)}(D)$ can be interpreted as the encoder transfer function relating input i to output j. As with any k-input, n-output linear system, there are a total of kn transfer functions. These can be represented by the $k \times n$ (transform domain) generator matrix

$$\mathbf{G}(D) = \begin{bmatrix} \mathbf{g}_1^{(0)}(D) & \mathbf{g}_1^{(1)}(D) & \cdots & \mathbf{g}_1^{(n-1)}(D) \\ \mathbf{g}_2^{(0)}(D) & \mathbf{g}_2^{(1)}(D) & \cdots & \mathbf{g}_2^{(n-1)}(D) \\ \vdots & \vdots & & \vdots \\ \mathbf{g}_k^{(0)}(D) & \mathbf{g}_k^{(1)}(D) & \cdots & \mathbf{g}_k^{(n-1)}(D) \end{bmatrix}. \tag{11.31}$$

Using the generator matrix, we can express the (transform domain) encoding equations for an (n, k, ν) feedforward encoder as

$$\mathbf{V}(D) = \mathbf{U}(D)\mathbf{G}(D), \tag{11.32}$$

where $\mathbf{U}(D) \triangleq [\mathbf{u}^{(1)}(D), \mathbf{u}^{(2)}(D), \cdots, \mathbf{u}^{(k)}(D)]$ is the k-tuple of input sequences, and $\mathbf{V}(D) \triangleq [\mathbf{v}^{(0)}(D), \mathbf{v}^{(1)}(D), \cdots, \mathbf{v}^{(n-1)}(D)]$ is the n-tuple of output sequences, or the codeword. After multiplexing, the codeword can also be expressed as

$$\mathbf{v}(D) = \mathbf{v}^{(0)}(D^n) + D\mathbf{v}^{(1)}(D^n) + \cdots + D^{n-1}\mathbf{v}^{(n-1)}(D^n). \tag{11.33}$$

EXAMPLE 11.2 (Continued)

For the (3, 2, 2) encoder of Figure 11.2,

$$\mathbf{G}(D) = \begin{bmatrix} 1+D & D & 1+D \\ D & 1 & 1 \end{bmatrix}. \tag{11.34}$$

For the input sequences $\mathbf{u}^{(1)}(D) = 1 + D^2$ and $\mathbf{u}^{(2)}(D) = 1 + D$, the encoding equations give the codeword

$$\begin{aligned} \mathbf{V}(D) = [\mathbf{v}^{(0)}(D), \mathbf{v}^{(1)}(D), \mathbf{v}^{(2)}(D)] &= [1 + D^2, 1 + D] \begin{bmatrix} 1+D & D & 1+D \\ D & 1 & 1 \end{bmatrix} \\ &= [1 + D^3, 1 + D^3, D^2 + D^3], \end{aligned} \tag{11.35a}$$

which also can be written as

$$\mathbf{v}(D) = 1 + D + D^8 + D^9 + D^{10} + D^{11}. \tag{11.35b}$$

Again, these results are the same as those calculated using convolution and matrix multiplication.

We can rewrite Eqs. (11.31), (11.32), and (11.33) to provide a means of representing the multiplexed codeword $\mathbf{v}(D)$ directly in terms of the input sequences. A little algebraic manipulation yields

$$\mathbf{v}(D) = \sum_{i=1}^{k} \mathbf{u}^{(i)}(D^n)\mathbf{g}_i(D), \tag{11.36}$$

where

$$\mathbf{g}_i(D) \triangleq \mathbf{g}_i^{(0)}(D^n) + D\mathbf{g}_i^{(1)}(D^n) + \cdots + D^{n-1}\mathbf{g}_i^{(n-1)}(D^n), \quad 1 \leq i \leq k, \tag{11.37}$$

is a *composite generator polynomial* relating the ith input sequence to $\mathbf{v}(D)$.

EXAMPLE 11.1 (Continued)

For the (2, 1, 3) encoder of Figure 11.1, the composite generator polynomial is

$$\mathbf{g}(D) = \mathbf{g}^{(0)}(D^2) + D\mathbf{g}^{(1)}(D^2) = 1 + D + D^3 + D^4 + D^5 + D^6 + D^7, \tag{11.38}$$

and for $\mathbf{u}(D) = 1 + D^2 + D^3 + D^4$, the codeword is

$$\begin{aligned}\mathbf{v}(D) &= \mathbf{u}(D^2)\mathbf{g}(D) = (1 + D^4 + D^6 + D^8)(1 + D + D^3 + D^4 + D^5 + D^6 + D^7) \\ &= 1 + D + D^3 + D^7 + D^9 + D^{11} + D^{14} + D^{15},\end{aligned} \tag{11.39}$$

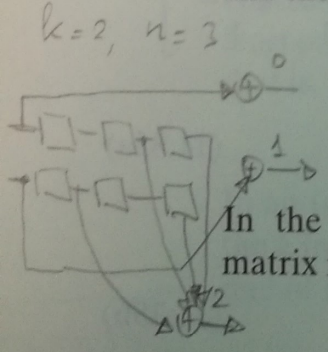
which again agrees with previous calculations.

An important subclass of convolutional encoders is the class of *systematic encoders*. In a systematic encoder, k output sequences, called *systematic output sequences*, are exact replicas of the k input sequences. Although, in general, the k systematic output sequences can be any set of k output sequences, we will follow the convention that the *first* k output sequences are the systematic sequences; that is,

$$\mathbf{v}^{(i-1)} = \mathbf{u}^{(i)}, \quad i = 1, 2, \cdots, k, \tag{11.40}$$

and the generator sequences satisfy

$$\mathbf{g}_i^{(j)} = \begin{cases} 1 \text{ if } j = i - 1 \\ 0 \text{ if } j \neq i - 1 \end{cases} \qquad i = 1, 2, \cdots, k. \tag{11.41}$$



In the case of *systematic feedforward encoders*, the (time-domain) generator matrix is given by

$$\mathbf{G} = \begin{bmatrix} \mathbf{IP}_0 & \mathbf{0P}_1 & \mathbf{0P}_2 & \cdots & \mathbf{0P}_m & & \\ & \mathbf{IP}_0 & \mathbf{0P}_1 & \cdots & \mathbf{0P}_{m-1} & \mathbf{0P}_m & \\ & & \mathbf{IP}_0 & \cdots & \mathbf{0P}_{m-2} & \mathbf{0P}_{m-1} & \mathbf{0P}_m \\ & & & \ddots & & \ddots & \end{bmatrix}, \tag{11.42}$$

where $\mathbf{I}$ is the $k \times k$ identity matrix, $\mathbf{0}$ is the $k \times k$ all-zero matrix, and $\mathbf{P}_l$ is the $k \times (n-k)$ matrix whose entries are

$$\mathbf{P}_l = \begin{bmatrix} g_{1,l}^{(k)} & g_{1,l}^{(k+1)} & \cdots & g_{1,l}^{(n-1)} \\ g_{2,l}^{k} & g_{2,l}^{(k+1)} & \cdots & g_{2,l}^{(n-1)} \\ \vdots & \vdots & & \vdots \\ g_{k,l}^{(k)} & g_{k,l}^{(k+1)} & \cdots & g_{k,l}^{(n-1)} \end{bmatrix}. \tag{11.43}$$

Similarly, the $k \times n$ (transform-domain) generator matrix becomes

$$\mathbf{G}(D) = \begin{bmatrix} 1 & 0 & \cdots & 0 & \mathbf{g}_1^{(k)}(D) & \cdots & \mathbf{g}_1^{(n-1)}(D) \\ 0 & 1 & \cdots & 0 & \mathbf{g}_2^{(k)}(D) & \cdots & \mathbf{g}_2^{(n-1)}(D) \\ \vdots & \vdots & & \vdots & \vdots & & \vdots \\ 0 & 0 & \cdots & 1 & \mathbf{g}_k^{(k)}(D) & \cdots & \mathbf{g}_k^{(n-1)}(D) \end{bmatrix}. \tag{11.44}$$

Because the first k output sequences are systematic, that is, they equal the k input sequences, they are also called *output information sequences*, and the last $n-k$ output sequences are called *output parity sequences*. Note that whereas, in general, kn generator polynomials must be specified to define an (n, k, ν) nonsystematic feedforward encoder, only $k(n-k)$ polynomials must be specified to define a systematic feedforward encoder. Hence, systematic encoders represent a subclass of the set of all possible encoders. Any encoder whose (transform-domain) generator matrix $\mathbf{G}(D)$ does not contain an identity matrix in some set of k columns is said to be *nonsystematic*.

For codes represented by systematic feedforward encoders with generator matrices in the form of (11.42) or (11.44), we can identify corresponding systematic parity-check matrices in a straightforward manner. The (time-domain) parity-check matrix is given by

$$\mathbf{H} = \begin{bmatrix} \mathbf{P}_0^T & \mathbf{I} & & & & & & & & & \\ \mathbf{P}_1^T & \mathbf{0} & \mathbf{P}_0^T & \mathbf{I} & & & & & & & \\ \mathbf{P}_2^T & \mathbf{0} & \mathbf{P}_1^T & \mathbf{0} & \mathbf{P}_0^T & \mathbf{I} & & & & & \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \ddots & & & & \\ \mathbf{P}_m^T & \mathbf{0} & \mathbf{P}_{m-1}^T & \mathbf{0} & \mathbf{P}_{m-2}^T & \mathbf{0} & \cdots & \mathbf{P}_0^T & \mathbf{I} & & \\ & & \mathbf{P}_m^T & \mathbf{0} & \mathbf{P}_{m-1}^T & \mathbf{0} & \cdots & \mathbf{P}_1^T & \mathbf{0} & \mathbf{P}_0^T & \mathbf{I} \\ & & & & \mathbf{P}_m^T & \mathbf{0} & \cdots & \mathbf{P}_2^T & \mathbf{0} & \mathbf{P}_1^T & \mathbf{0} \\ & & & & & & & & & & \ddots \end{bmatrix}, \tag{11.45}$$

where, in this case, $\mathbf{I}$ is the $(n-k) \times (n-k)$ identity matrix, and $\mathbf{0}$ is the $(n-k) \times (n-k)$ all-zero matrix. Similarly, the $(n-k) \times n$ (transform-domain) parity-check matrix is

given by

$$\mathbf{H}(D) = \begin{bmatrix} \mathbf{g}_1^{(k)}(D) & \mathbf{g}_2^{(k)}(D) & \cdots & \mathbf{g}_k^{(k)}(D) & 1 & 0 & \cdots & 0 \\ \mathbf{g}_1^{(k+1)}(D) & \mathbf{g}_2^{(k+1)}(D) & \cdots & \mathbf{g}_k^{(k+1)}(D) & 0 & 1 & \cdots & 0 \\ \vdots & \vdots & & \vdots & \vdots & \vdots & & \vdots \\ \mathbf{g}_1^{(n-1)}(D) & \mathbf{g}_2^{(n-1)}(D) & \cdots & \mathbf{g}_k^{(n-1)}(D) & 0 & 0 & \cdots & 1 \end{bmatrix}, \quad (11.46a)$$

where the last $(n-k)$ columns of $\mathbf{H}(D)$ form the $(n-k) \times (n-k)$ identity matrix. The parity check matrix in (11.46a) can be rewritten as

$$\mathbf{H}(D) = \begin{bmatrix} \mathbf{h}_1^{(0)}(D) & \mathbf{h}_1^{(1)}(D) & \cdots & \mathbf{h}_1^{(k-1)}(D) & 1 & 0 & \cdots & 0 \\ \mathbf{h}_2^{(0)}(D) & \mathbf{h}_2^{(1)}(D) & \cdots & \mathbf{h}_2^{(k-1)}(D) & 0 & 1 & \cdots & 0 \\ \vdots & \vdots & & \vdots & \vdots & \vdots & & \vdots \\ \mathbf{h}_{n-k}^{(0)}(D) & \mathbf{h}_{n-k}^{(1)}(D) & \cdots & \mathbf{h}_{n-k}^{(k-1)}(D) & 0 & 0 & \cdots & 1 \end{bmatrix}, \quad (11.46b)$$

where we have defined the $(j-k+1)$st parity-check polynomial associated with the $(i-1)$st output information sequence as $\mathbf{h}_{j-k+1}^{(i-1)}(D) = \mathbf{g}_i^{(j)}(D)$, $j = k, k+1, \cdots, n-1$, and $i = 1, 2, \cdots, k$. Simplifying notation, we can write the $(n-k) \times k$ parity check polynomials as $\mathbf{h}_i^{(j)}(D)$, $i = 1, 2, \cdots, n-k$, and $j = 0, 1, \cdots, k-1$. (In the important special case of rate $R = (n-1)/n$ codes, i.e., $(n-k) = 1$, we denote the $k = n-1$ parity-check polynomials without subscripts as $\mathbf{h}^{(j)}(D)$, $j = 0, 1, \cdots, k-1$.) Finally, we associate the parity-check sequence $h_i^{(j)}$ with the coefficients of the parity-check polynomial $h_i^{(j)}(D)$.

Any codeword $\mathbf{v}$ (or $\mathbf{V}(D)$) must then satisfy the parity-check equations

$$\mathbf{v}\mathbf{H}^T = \mathbf{0}, \quad (11.47)$$

where the matrix $\mathbf{0}$ has semi-infinite dimensions in this case, or

$$\mathbf{V}(D)\mathbf{H}^T(D) = \mathbf{0}(D), \quad (11.48)$$

where $\mathbf{0}(D)$ represents the $1 \times (n-k)$ matrix of all-zero sequences. In other words, the set of all codewords $\mathbf{v}$ (or $\mathbf{V}(D)$) is the null space of the parity-check matrix $\mathbf{H}$ (or $\mathbf{H}(D)$).

EXAMPLE 11.4 A Rate $R = 1/2$ Systematic Feedforward Convolutional Encoder

Consider the $(2, 1, 3)$ systematic feedforward encoder shown in Figure 11.4. The generator sequences are $\mathbf{g}^{(0)} = (1\ 0\ 0\ 0)$ and $\mathbf{g}^{(1)} = (1\ 1\ 0\ 1)$, and the generator matrices are given by

$$\mathbf{G} = \begin{bmatrix} 11 & 01 & 00 & 01 & & \\ & 11 & 01 & 00 & 01 & \\ & & 11 & 01 & 00 & 01 \\ & & & \ddots & & \ddots \end{bmatrix}, \quad (11.49)$$

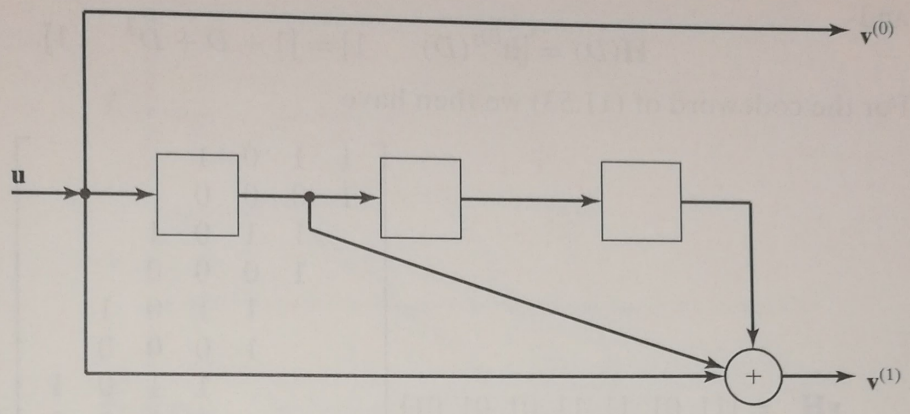


FIGURE 11.4: A (2, 1, 3) binary systematic feedforward convolutional encoder.

and

$$\mathbf{G}(D) = [1 \quad 1 + D + D^3]. \tag{11.50}$$

For an input sequence $\mathbf{u}(D) = 1 + D^2 + D^3$, the output information sequence is

$$\mathbf{v}^{(0)}(D) = \mathbf{u}(D)\mathbf{g}^{(0)}(D) = (1 + D^2 + D^3)(1) = 1 + D^2 + D^3, \tag{11.51}$$

the output parity sequence is

$$\begin{aligned}\mathbf{v}^{(1)}(D) &= \mathbf{u}(D)\mathbf{g}^{(1)}(D) = (1 + D^2 + D^3)(1 + D + D^3) \\ &= 1 + D + D^2 + D^3 + D^4 + D^5 + D^6,\end{aligned} \tag{11.52}$$

and the codeword is given by

$$\mathbf{V}(D) = [1 + D^2 + D^3 \quad 1 + D + D^2 + D^3 + D^4 + D^5 + D^6] \tag{11.53a}$$

or

$$\begin{aligned}\mathbf{v}(D) &= \mathbf{v}^{(0)}(D^2) + D\mathbf{v}^{(1)}(D^2) \\ &= 1 + D + D^3 + D^4 + D^5 + D^6 + D^7 + D^9 + D^{11} + D^{13},\end{aligned} \tag{11.53b}$$

which we can also write as

$$\mathbf{v} = (11, 01, 11, 11, 01, 01, 01). \tag{11.53c}$$

Using (11.45) and (11.46b), we see that the parity-check matrices are given by

$$\mathbf{H} = \begin{bmatrix} 11 & & & & & & \\ 10 & 11 & & & & & \\ 00 & 10 & 11 & & & & \\ 10 & 00 & 10 & 11 & & & \\ & 10 & 00 & 10 & 11 & & \\ & & 10 & 00 & 10 & 11 & \\ & & & \ddots & & \ddots & \end{bmatrix} \tag{11.54}$$

and

$$\mathbf{H}(D) = [\mathbf{h}^{(0)}(D) \quad 1] = [1 + D + D^3 \quad 1]. \tag{11.55}$$

For the codeword of (11.53) we then have

$$\mathbf{v}\mathbf{H}^T = [11, 01, 11, 11, 01, 01, 01] \begin{bmatrix} 1 & 1 & 0 & 1 & & & & \\ 1 & 0 & 0 & 0 & & & & \\ & 1 & 1 & 0 & 1 & & & \\ & 1 & 0 & 0 & 0 & & & \\ & & 1 & 1 & 0 & 1 & & \\ & & 1 & 0 & 0 & 0 & & \\ & & & 1 & 1 & 0 & 1 & \\ & & & 1 & 0 & 0 & 0 & \\ & & & & 1 & 1 & 0 & \\ & & & & 1 & 0 & 0 & \\ & & & & & 1 & 1 & \\ & & & & & 1 & 0 & \\ & & & & & & 1 & \\ & & & & & & 1 & \end{bmatrix} = \mathbf{0}, \tag{11.56}$$

or

$$\mathbf{V}(D)\mathbf{H}^T(D) = [1 + D^2 + D^3 \quad 1 + D + D^2 + D^3 + D^4 + D^5 + D^6] \begin{bmatrix} 1 + D + D^3 \\ 1 \end{bmatrix}$$

$$= \mathbf{0}(D). \tag{11.57}$$

EXAMPLE 11.5 A Rate $R = 2/3$ Systematic Feedforward Convolutional Encoder

Now, consider the rate $R = 2/3$ systematic feedforward encoder with generator matrix

$$\mathbf{G}(D) = \begin{bmatrix} 1 & 0 & 1 + D + D^2 \\ 0 & 1 & 1 + D \end{bmatrix}. \tag{11.58}$$

Using (11.46b), we can write the parity-check matrix as

$$\mathbf{H}(D) = \begin{bmatrix} \mathbf{h}^{(0)}(D) & \mathbf{h}^{(1)}(D) & 1 \end{bmatrix} = \begin{bmatrix} 1 + D + D^2 & 1 + D & 1 \end{bmatrix}. \tag{11.59}$$

The controller canonical form realization of the encoder requires a total of $\nu = \nu_1 + \nu_2 = 3$ delay elements and is shown in Figure 11.5(a). This realization results in a $(3, 2, 3)$ encoder. Moreover, since the output information sequences are given by $\mathbf{v}^{(0)}(D) = \mathbf{u}^{(1)}(D)$ and $\mathbf{v}^{(1)}(D) = \mathbf{u}^{(2)}(D)$, and the output parity sequence is given by

$$\begin{aligned} \mathbf{v}^{(2)}(D) &= \mathbf{u}^{(1)}(D)\mathbf{g}_1^{(2)}(D) + \mathbf{u}^{(2)}(D)\mathbf{g}_2^{(2)}(D), \\ &= (1 + D + D^2)\mathbf{u}^{(1)}(D) + (1 + D)\mathbf{u}^{(2)}(D), \end{aligned} \tag{11.60}$$

the encoder can also be realized as shown in Figure 11.5(b). Note that this realization requires only two delay elements rather than three.

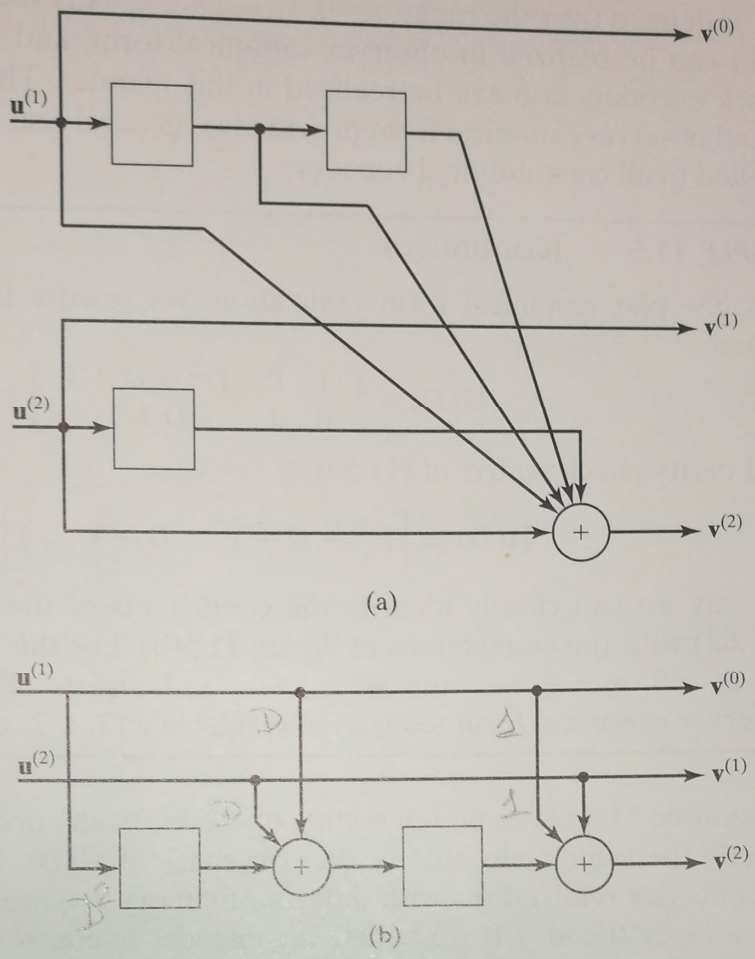


FIGURE 11.5: (a) Controller canonical form and (b) observer canonical form realizations of a rate $R = 2/3$ binary systematic feedforward convolutional encoder.

The encoder realization of Figure 11.5(b) is called *observer canonical form*. In general, for observer canonical form encoder realizations, there is one shift register corresponding to each of the n output sequences, the k input sequences enter modulo-2 adders internal to the shift registers, and the outputs at the right end of each shift register form the n output sequences. Also, the lowest-degree (constant) terms in the generator (parity-check) polynomials represent the connections at the right ends of the shift registers, whereas the highest-degree terms represent the connections at the left ends of the shift registers. It is important to note that this is exactly the opposite of the correspondence between polynomial coefficients and delay elements in the case of a controller canonical form realization. For this reason, when an encoder is realized in observer canonical form, it is common to write the generator (parity-check) polynomials in the opposite of the usual order, that is, from highest degree to lowest degree. In the case of systematic encoders like the one in Figure 11.5(b), there are only $(n - k)$ shift registers, and the k input sequences appear directly as the first k encoder output sequences. Definitions related to the lengths of the shift registers used in convolutional encoders realized in observer canonical form can be given that are analogous to Definitions 11.1–11.3. The only difference is that in this case there are, in general, n shift registers rather then k,

and ν_j is defined over the range $j = 0, 1, \cdots, n-1$. Any feedforward convolutional encoder can be realized in observer canonical form, and we will see shortly that feedback encoders also can be realized in this manner. Thus, controller canonical form and observer canonical form provide two general realization methods that can be applied to all convolutional encoders.

EXAMPLE 11.5 (Continued)

For an observer canonical form realization, we rewrite the generator matrix of (11.58) as

$$\mathbf{G}(D) = \begin{bmatrix} 1 & 0 & D^2 + D + 1 \\ 0 & 1 & D + 1 \end{bmatrix} \tag{11.61}$$

and the parity-check matrix of (11.59) as

$$\mathbf{H}(D) = \begin{bmatrix} D^2 + D + 1 & D + 1 & 1 \end{bmatrix}. \tag{11.62}$$

In this way we can clearly identify the coefficients of the polynomials in (11.61) and (11.62) with the connections in Figure 11.5(b). For this realization, we see that $\nu_0 = 0$, $\nu_1 = 0$, $\nu_2 = 2$, $m = \max_{(0 \le j \le n-1)} \nu_j = 2$, and $\nu = \Sigma_{(0 \le j \le n-1)} \nu_j = 2$. Thus, the observer canonical form realization results in a $(3, 2, 2)$ encoder.

Example 11.5 raises an interesting question. Both encoder realizations generate exactly the same code, that is, the row space of $\mathbf{G}(D)$, but since there are two distinct encoder realizations with different memory requirements, is this a $(3, 2, 3)$ code or a $(3, 2, 2)$ code? If we realize the encoder in controller canonical form, the overall constraint length is $\nu = 3$, resulting in a $(3, 2, 3)$ encoder; however, if we realize the encoder in observer canonical form, then $\nu = 2$, resulting in a $(3, 2, 2)$ encoder. We will see in the next section that since a convolutional encoder can be realized as a linear sequential circuit, it can be described using a state diagram with 2^ν states. Also, we will see in the next chapter that maximum likelihood decoding and maximum a posteriori probability decoding of convolutional codes require a decoding complexity that is proportional to the number of states in the encoder state diagram. Hence, we always seek an encoder realization with a minimal number of states, that is, with a minimal overall constraint length ν.

We will continue our discussion of *minimal encoder realizations* after introducing another class of systematic encoders. *Systematic feedback encoders* generate the same codes as corresponding nonsystematic feedforward encoders but exhibit a different mapping between information sequences and codewords. We again illustrate the basic concepts using several examples.

EXAMPLE 11.6 A Rate $R = 1/3$ Systematic Feedback Convolutional Encoder

Consider the rate $R = 1/3$ nonsystematic feedforward convolutional encoder with generator matrix

$$\mathbf{G}(D) = \begin{bmatrix} \mathbf{g}^{(0)}(D) & \mathbf{g}^{(1)}(D) & \mathbf{g}^{(2)}(D) \end{bmatrix} = \begin{bmatrix} 1 + D + D^2 & 1 + D^2 & 1 + D \end{bmatrix}. \tag{11.63a}$$

The (3, 1, 2) controller canonical form realization is shown in Figure 11.6(a). Reversing the order of the polynomials in (11.63a), we have

$$\mathbf{G}(D) = \begin{bmatrix} D^2 + D + 1 & D^2 + 1 & D + 1 \end{bmatrix}, \tag{11.63b}$$

from which we obtain the (3, 1, 5) observer canonical form encoder realization shown in Figure 11.6(b). Note that in this case the controller canonical form realization has overall constraint length $\nu = 2$, that is, 4 states, whereas the observer canonical form realization requires 32 states! Now, we divide each entry in $\mathbf{G}(D)$ by the polynomial $\mathbf{g}^{(0)}(D) = 1 + D + D^2$ to obtain the modified generator matrix

$$\begin{aligned} \mathbf{G}'(D) &= [1 \quad \mathbf{g}^{(1)}(D)/\mathbf{g}^{(0)}(D) \quad \mathbf{g}^{(2)}(D)/\mathbf{g}^{(0)}(D)] \\ &= [1 \quad (1 + D^2)/(1 + D + D^2) \quad (1 + D)/(1 + D + D^2)] \end{aligned} \tag{11.64}$$

in systematic feedback form. We note in this case that the impulse response of the encoder represented by $\mathbf{G}'(D)$ has infinite duration; that is, the feedback shift register realization of $\mathbf{G}'(D)$ is an *infinite impulse response* (IIR) linear system. For this reason, the (time-domain) generator matrix $\mathbf{G}'$ corresponding to $\mathbf{G}'(D)$ contains sequences of infinite length. For example, the ratio of generator polynomials $\mathbf{g}^{(1)}(D)/\mathbf{g}^{(0)}(D)$ in (11.64) produces the power series $(1 + D^2)/(1 + D + D^2) = 1 + D + D^2 + D^4 + D^5 + D^7 + D^8 + D^{10} + \cdots$, whose time-domain equivalent is the infinite-length sequence $(11101101101\cdots)$.

The code, that is, the set of encoder output sequences (codewords), generated by $\mathbf{G}'(D)$ is exactly the same as the code generated by $\mathbf{G}(D)$. This can be seen by noting that if the information sequence $\mathbf{u}(D)$ produces the codeword $\mathbf{V}(D)$, that is, $\mathbf{V}(D) = \mathbf{u}(D)\mathbf{G}(D)$, in the code generated by $\mathbf{G}(D)$, then the information sequence $\mathbf{u}'(D) = (1 + D + D^2)\mathbf{u}(D)$ produces the same codeword $\mathbf{V}(D)$, that is, $\mathbf{V}(D) = \mathbf{u}'(D)\mathbf{G}'(D)$, in the code generated by $\mathbf{G}'(D)$. In other words, the row spaces of $\mathbf{G}(D)$ and $\mathbf{G}'(D)$ are the same, and thus they generate the same code. It is important to note, however, that there is a different mapping between information sequences and codewords in the two cases. The 4-state, (3, 1, 2) encoder diagram, in controller canonical form, corresponding to the systematic feedback encoder of (11.64), is shown in Figure 11.6(c). In contrast, the observer canonical form realization of $\mathbf{G}'(D)$ requires 16 states (see Problem 11.8).

Finally, since $\mathbf{G}'(D)$ is systematic, we can use (11.46a) to obtain the parity-check matrix

$$\mathbf{H}'(D) = \begin{bmatrix} (1 + D^2)/(1 + D + D^2) & 1 & 0 \\ (1 + D)/(1 + D + D^2) & 0 & 1 \end{bmatrix}, \tag{11.65}$$

where $\mathbf{H}'(D)$ is a valid parity-check matrix for both $\mathbf{G}(D)$ and $\mathbf{G}'(D)$; that is, all codewords in the row spaces of $\mathbf{G}(D)$ and $\mathbf{G}'(D)$ are also in the null space of $\mathbf{H}'(D)$, and as noted for generator sequences, the systematic (time-domain) parity-check matrix $\mathbf{H}'$ corresponding to $\mathbf{G}'$ contains sequences of infinite length.

In general, if the $k \times n$ polynomial matrix $\mathbf{G}(D)$ is the generator matrix of a rate $R = k/n$ nonsystematic convolutional encoder, then elementary (polynomial) row operations can be performed on $\mathbf{G}(D)$ to convert it to a systematic $k \times n$

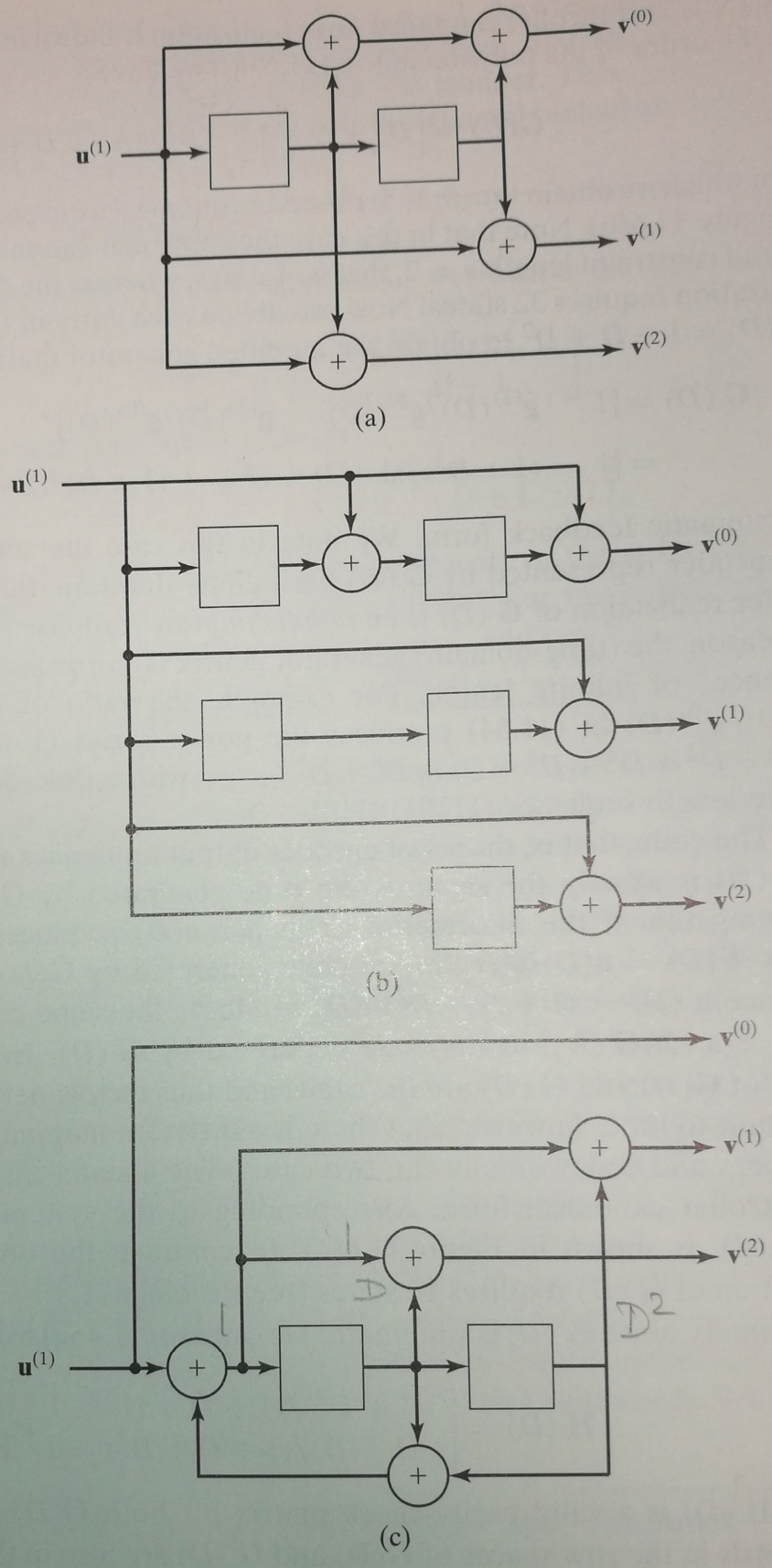


FIGURE 11.6: (a) The controller canonical form and (b) the observer canonical form realizations of a nonsystematic feedforward encoder, and (c) the controller canonical form realization of an equivalent systematic feedback encoder.

generator matrix $\mathbf{G}'(D)$, whose entries are *rational functions* in the delay operator D and whose first k columns form the $k \times k$ identity matrix. Because elementary row operations do not change the row space of a matrix, the matrices $\mathbf{G}(D)$ and $\mathbf{G}'(D)$ generate the same code, and since $\mathbf{G}'(D)$ contains rational functions, it results in a feedback encoder realization. Also, since $\mathbf{G}'(D)$ is in systematic form, it can be used to determine an $(n-k) \times n$ (transform-domain) systematic parity-check matrix $\mathbf{H}'(D)$ in the same way that we used (11.46a) to find a parity-check matrix for a systematic feedforward encoder. In this case, $\mathbf{H}'(D)$ contains rational functions, and its last $(n-k)$ columns form the $(n-k) \times (n-k)$ identity matrix. Because the code is the null space of $\mathbf{H}'(D)$, and since elementary row operations do not change the null space of a matrix, we can convert $\mathbf{H}'(D)$ to an equivalent nonsystematic parity-check matrix $\mathbf{H}(D)$ containing only polynomial entries.

That the generator and parity-check sequences in Example 11.6 exhibit IIR behavior is characteristic of feedback encoders. For this reason, feedback encoders are more easily described using the (transform-domain) matrices $\mathbf{G}'(D)$ and $\mathbf{H}'(D)$, whose entries contain rational functions, rather than the corresponding (time-domain) matrices $\mathbf{G}'$ and $\mathbf{H}'$, whose entries contain sequences of infinite length. Because the response to a single nonzero input in a feedback encoder has infinite duration, the codes produced by feedback encoders are called *recursive convolutional codes*. Thus, we say that a systematic feedback encoder produces a *systematic recursive convolutional code* (SRCC). It is important to note that the SRCC produced by a systematic feedback encoder contains exactly the same set of codewords as the nonrecursive convolutional code produced by an equivalent nonsystematic feedforward encoder; however, as noted in Example 11.6, the mapping between information sequences and codewords is different in the two cases. In particular, for nonrecursive codes, information sequences of weight 1 produce finite-length codewords, whereas for recursive codes, information sequences of weight 1 produce codewords of infinite length. This property of recursive codes turns out to be a key factor in the excellent performance achieved by the class of parallel concatenated convolutional codes, or turbo codes, to be discussed in Chapter 16.

EXAMPLE 11.2 (Continued)

The nonsystematic feedforward generator matrix $\mathbf{G}(D)$ in Example 11.2 was given in (11.34), and the 4-state, (3, 2, 2) controller canonical form encoder realization was shown in Figure 11.2. (Note that the observer canonical form realization of (11.34) would require 8 states.) To convert $\mathbf{G}(D)$ to an equivalent systematic feedback encoder, we apply the following sequence of elementary row operations:

Step 1. Row $1 \Rightarrow [1/(1+D)][\text{Row } 1]$.
Step 2. Row $2 \Rightarrow Row2 + [D][\text{Row } 1]$.
Step 3. Row $2 \Rightarrow [(1+D)/(1+D+D^2)][\text{Row } 2]$.
Step 4. Row $1 \Rightarrow Row\ 1 + [D/(1+D)][\text{Row } 2]$.

The result is the modified generator matrix (see Problem 11.7)

$$\mathbf{G}'(D) = \begin{bmatrix} 1 & 0 & 1/(1+D+D^2) \\ 0 & 1 & (1+D^2)/(1+D+D^2) \end{bmatrix} \tag{11.66}$$

in systematic feedback form. Now, we can use (11.46b) to form the systematic parity-check matrix

$$\begin{aligned}\mathbf{H}'(D) &= [\mathbf{h}^{(0)}(D)/\mathbf{h}^{(2)}(D) \quad \mathbf{h}^{(1)}(D)/\mathbf{h}^{(2)}(D) \quad 1] \\ &= [1/(1+D+D^2) \quad (1+D^2)/(1+D+D^2) \quad 1]\end{aligned} \tag{11.67a}$$

or the equivalent nonsystematic polynomial parity-check matrix

$$\mathbf{H}(D) = [\mathbf{h}^{(0)}(D) \quad \mathbf{h}^{(1)}(D) \quad \mathbf{h}^{(2)}(D)] = [1 \quad 1+D^2 \quad 1+D+D^2], \tag{11.67b}$$

where $\mathbf{h}^{(j)}(D)$ represents the parity-check polynomial associated with the jth output sequence. Again, the recursive code generated by $\mathbf{G}'(D)$ is exactly the same as the nonrecursive code generated by $\mathbf{G}(D)$, and the encoder diagram, in controller canonical form, corresponding to the systematic feedback encoder of (11.66), is shown in Figure 11.7(a). We note in this case that Figure 11.7(a) represents a 16-state, $(3, 2, 4)$ encoder; that is, two length-2 shift registers are needed in the controller canonical form realization. Reversing the order of the polynomials in (11.66) and (11.67a), we obtain

$$\mathbf{G}'(D) = \begin{bmatrix} 1 & 0 & 1/(D^2+D+1) \\ 0 & 1 & (D^2+1)/(D^2+D+1) \end{bmatrix} \tag{11.68}$$

and

$$\mathbf{H}'(D) = [1/(D^2+D+1) \quad (D^2+1)/(D^2+D+1) \quad 1], \tag{11.69}$$

which leads to the observer canonical form systematic feedback encoder realization shown in Figure 11.7(b). We note that this represents a 4-state, $(3, 2, 2)$ encoder; that is, for the systematic feedback encoder, the observer canonical form realization requires fewer states than the controller canonical form realization.

The rate $R = 2/3$ code of Example 11.2 has a 4-state nonsystematic feedforward encoder realization in controller canonical form (see Figure 11.2). It also has a 4-state systematic feedback encoder realization in observer canonical form (see Figure 11.7(b)). Thus, in this case, both encoder realizations have an overall constraint length $\nu = 2$; that is, we obtain a minimal encoder realization in both controller canonical form and observer canonical form. We now give an example of a rate $R = 2/3$ code in which a minimal encoder realization is obtained only in observer canonical form.

EXAMPLE 11.7 A Rate $R = 2/3$ Systematic Feedback Convolutional Encoder

Now, consider the rate $R = 2/3$ nonsystematic feedforward generator matrix given by

$$\mathbf{G}(D) = \begin{bmatrix} 1 & D & 1+D \\ 0 & 1+D+D^2+D^3 & 1+D^2+D^3 \end{bmatrix}. \tag{11.70}$$

A 16-state, $(3, 2, 4)$ controller canonical form encoder realization is shown in Figure 11.8(a). (Note that the observer canonical form realization of (11.70) would

in systematic feedback form. Now, we can use (11.46b) to form the systematic parity-check matrix

$$\begin{aligned}\mathbf{H}'(D) &= [\mathbf{h}^{(0)}(D)/\mathbf{h}^{(2)}(D) \quad \mathbf{h}^{(1)}(D)/\mathbf{h}^{(2)}(D) \quad 1] \\ &= [1/(1+D+D^2) \quad (1+D^2)/(1+D+D^2) \quad 1]\end{aligned} \tag{11.67a}$$

or the equivalent nonsystematic polynomial parity-check matrix

$$\mathbf{H}(D) = [\mathbf{h}^{(0)}(D) \quad \mathbf{h}^{(1)}(D) \quad \mathbf{h}^{(2)}(D)] = [1 \quad 1+D^2 \quad 1+D+D^2], \tag{11.67b}$$

where $\mathbf{h}^{(j)}(D)$ represents the parity-check polynomial associated with the jth output sequence. Again, the recursive code generated by $\mathbf{G}'(D)$ is exactly the same as the nonrecursive code generated by $\mathbf{G}(D)$, and the encoder diagram, in controller canonical form, corresponding to the systematic feedback encoder of (11.66), is shown in Figure 11.7(a). We note in this case that Figure 11.7(a) represents a 16-state, $(3, 2, 4)$ encoder; that is, two length-2 shift registers are needed in the controller canonical form realization. Reversing the order of the polynomials in (11.66) and (11.67a), we obtain

$$\mathbf{G}'(D) = \begin{bmatrix} 1 & 0 & 1/(D^2+D+1) \\ 0 & 1 & (D^2+1)/(D^2+D+1) \end{bmatrix} \tag{11.68}$$

and

$$\mathbf{H}'(D) = [1/(D^2+D+1) \quad (D^2+1)/(D^2+D+1) \quad 1], \tag{11.69}$$

which leads to the observer canonical form systematic feedback encoder realization shown in Figure 11.7(b). We note that this represents a 4-state, $(3, 2, 2)$ encoder; that is, for the systematic feedback encoder, the observer canonical form realization requires fewer states than the controller canonical form realization.

The rate $R = 2/3$ code of Example 11.2 has a 4-state nonsystematic feedforward encoder realization in controller canonical form (see Figure 11.2). It also has a 4-state systematic feedback encoder realization in observer canonical form (see Figure 11.7(b)). Thus, in this case, both encoder realizations have an overall constraint length $\nu = 2$; that is, we obtain a minimal encoder realization in both controller canonical form and observer canonical form. We now give an example of a rate $R = 2/3$ code in which a minimal encoder realization is obtained only in observer canonical form.

EXAMPLE 11.7 A Rate $R = 2/3$ Systematic Feedback Convolutional Encoder

Now, consider the rate $R = 2/3$ nonsystematic feedforward generator matrix given by

$$\mathbf{G}(D) = \begin{bmatrix} 1 & D & 1+D \\ 0 & 1+D+D^2+D^3 & 1+D^2+D^3 \end{bmatrix}. \tag{11.70}$$

A 16-state, $(3, 2, 4)$ controller canonical form encoder realization is shown in Figure 11.8(a). (Note that the observer canonical form realization of (11.70) would

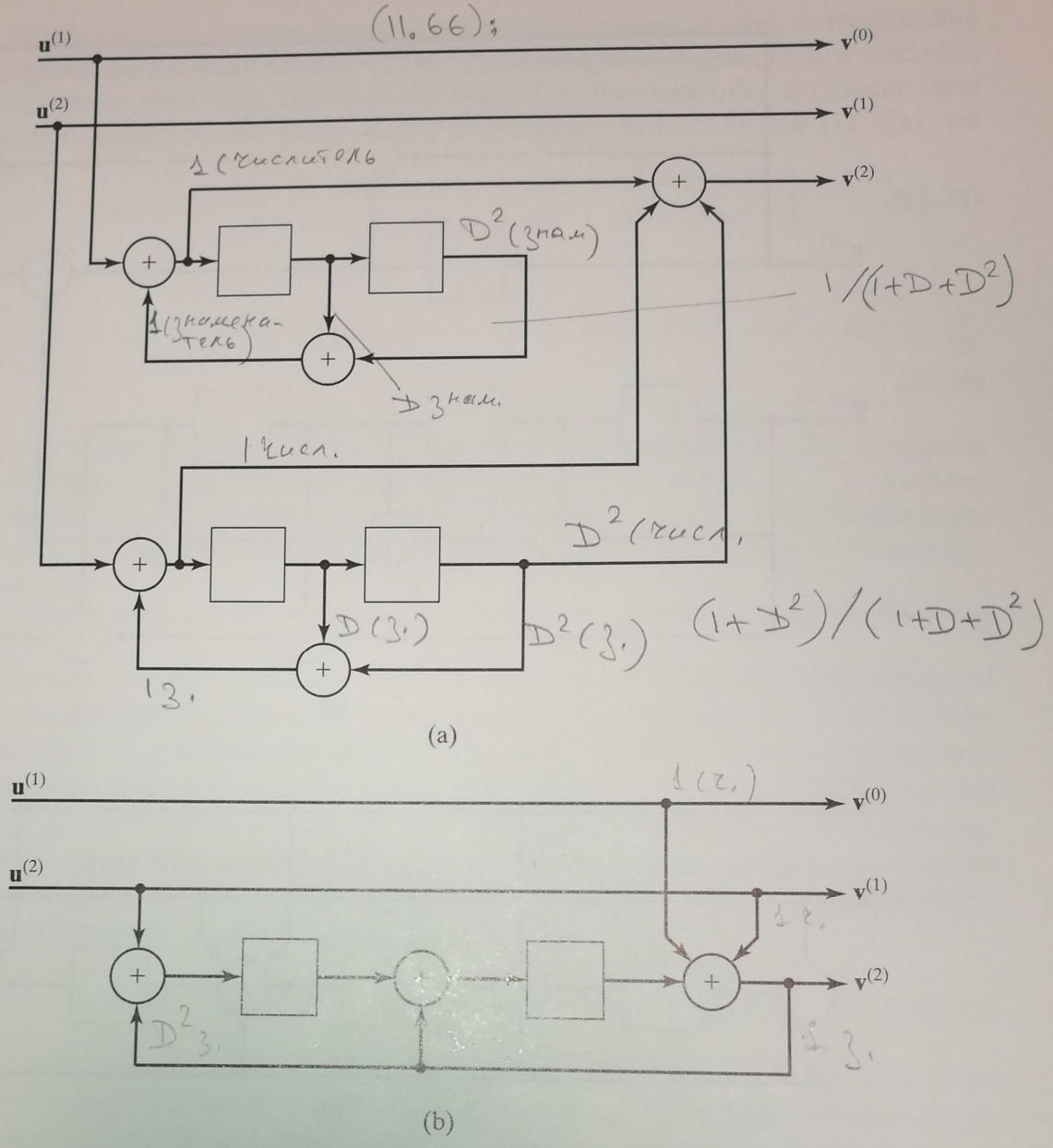


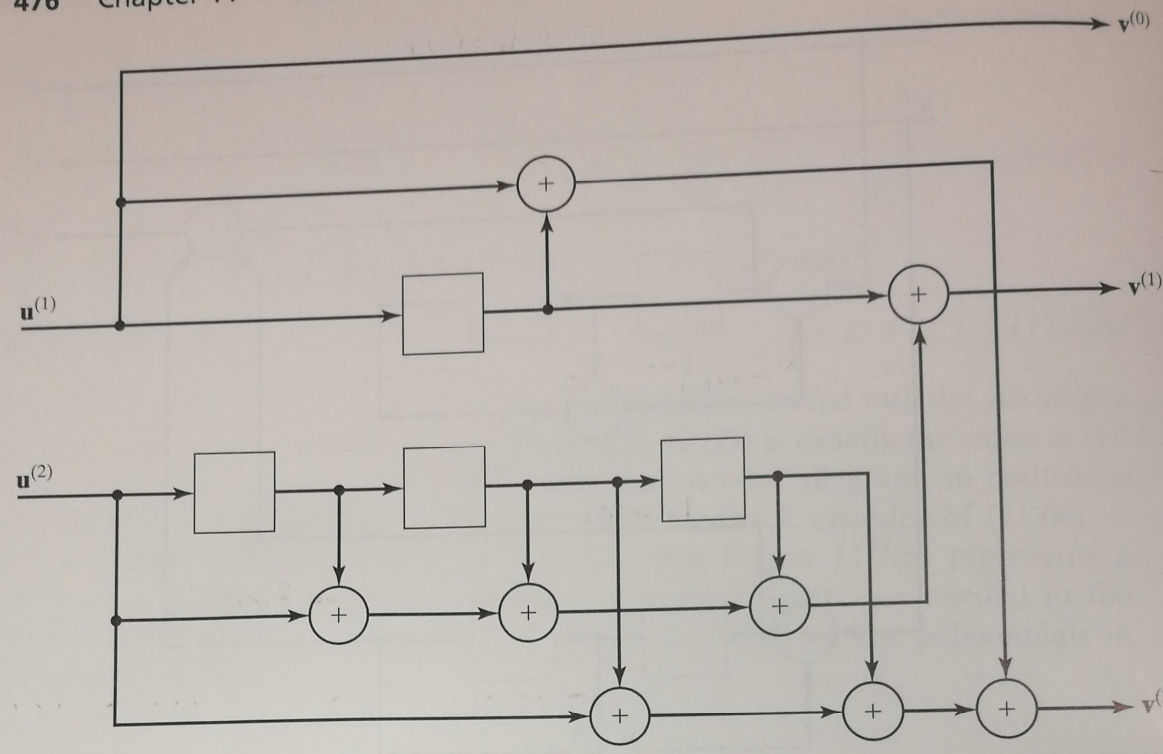
FIGURE 11.7: (a) A (3, 2, 4) systematic feedback encoder in controller canonical form and (b) an equivalent (3, 2, 2) systematic feedback encoder in observer canonical form.

require 64 states.) To convert $\mathbf{G}(D)$ to an equivalent systematic feedback encoder, we apply the following sequence of elementary row operations:

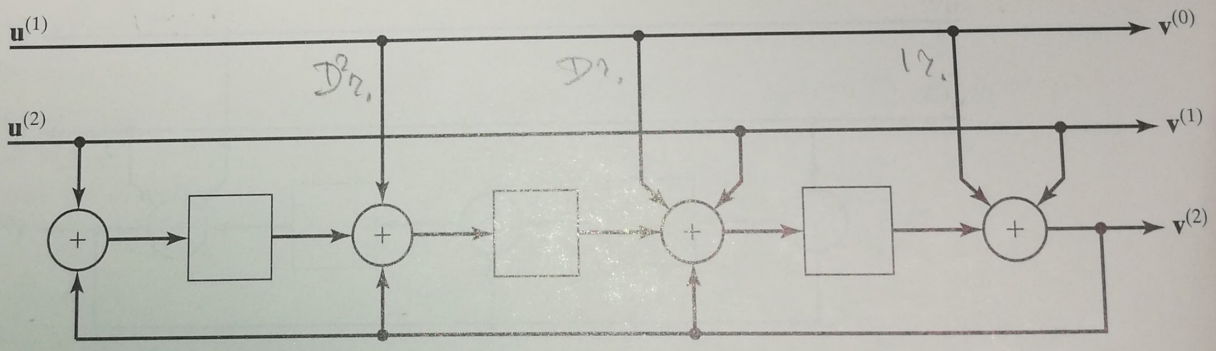
Step 1. Row 2 $\Rightarrow [1/(1 + D + D^2 + D^3)]$ [Row 2].
Step 2. Row 1 $\Rightarrow$ Row 1 + [D][Row 2].

The result is the modified generator matrix (see Problem 11.7)

$$\mathbf{G}'(D) = \begin{bmatrix} 1 & 0 & (1 + D + D^2)/(1 + D + D^2 + D^3) \\ 0 & 1 & (1 + D + D^3)/(1 + D + D^2 + D^3) \end{bmatrix} \tag{11.71}$$



(a)



(b)

FIGURE 11.8: (a) A (3, 2, 4) nonsystematic feedforward encoder in controller canonical form and (b) an equivalent (3, 2, 3) systematic feedback encoder in observer canonical form.

in systematic feedback form, and we can use (11.46b) to form the systematic parity-check matrix

$$\begin{aligned}\mathbf{H}'(D) &= [\mathbf{h}^{(0)}(D)/\mathbf{h}^{(2)}(D) \quad \mathbf{h}^{(1)}(D)/\mathbf{h}^{(2)}(D) \quad 1] \\ &= [(1+D+D^2)/(1+D+D^2+D^3) \quad (1+D+D^3)/(1+D+D^2+D^3) \quad 1]\end{aligned} \tag{11.72a}$$

or the equivalent nonsystematic polynomial parity-check matrix

$$\mathbf{H}(D) = [1+D+D^2 \quad 1+D+D^3 \quad 1+D+D^2+D^3]. \tag{11.72b}$$

In this case, the encoder diagram, in controller canonical form, corresponding to the systematic feedback encoder of (11.71), represents a 64-state, (3, 2, 6) encoder; that is, two length-3 shift registers are needed in the controller canonical form realization. Reversing the order of the polynomials in (11.71) and (11.72a), we obtain

$$\mathbf{G}'(D) = \begin{bmatrix} 1 & 0 & (D^2+D+1)/(D^3+D^2+D+1) \\ 0 & 1 & (D^3+D+1)/(D^3+D^2+D+1) \end{bmatrix} \tag{11.73}$$

and

$$\mathbf{H}'(D) = [(D^2+D+1)/(D^3+D^2+D+1) \quad (D^3+D+1)/(D^3+D^2+D+1) \quad 1], \tag{11.74}$$

which leads to the observer canonical form systematic feedback encoder realization shown in Figure 11.8(b). We note that this represents an 8-state, (3, 2, 3) encoder; that is, the observer canonical form systematic feedback encoder realization requires fewer states than the controller canonical form nonsystematic feedforward encoder realization.

Example 11.6 shows that for a rate $R = 1/3$ code, a minimal encoder realization is obtained in controller canonical form. In general, for any rate $R = 1/n$ encoder described by a $1 \times n$ polynomial generator matrix $\mathbf{G}(D)$ or an equivalent systematic rational function generator matrix $\mathbf{G}'(D)$, the controller canonical form realization of either $\mathbf{G}(D)$ or $\mathbf{G}'(D)$ is minimal, provided all common factors have been removed from $\mathbf{G}(D)$. (Removing common factors from a $1 \times n$ polynomial generator matrix does not change its row space.) For example, consider the rate $R = 1/2$ nonsystematic feedforward encoder described by the generator matrix

$$\mathbf{G}(D) = [\mathbf{g}^{(0)}(D) \quad \mathbf{g}^{(1)}(D)] = [1+D^2 \quad 1+D^3]. \tag{11.75}$$

The controller canonical form realization of this encoder has overall constraint length $\nu = 3$; that is, it has $2^\nu = 8$ states; however, if we remove the common factor $(1+D)$ from the generator polynomials $\mathbf{g}^{(0)}(D)$ and $\mathbf{g}^{(1)}(D)$, we obtain the generator matrix

$$\mathbf{G}(D) = [\mathbf{g}^{(0)}(D) \quad \mathbf{g}^{(1)}(D)] = [1+D \quad 1+D+D^2] \tag{11.76a}$$

or its systematic feedback equivalent,

$$\mathbf{G}'(D) = [1 \quad \mathbf{g}^{(1)}(D)/\mathbf{g}^{(0)}(D)] = [1 \quad (1+D+D^2)/(1+D)]. \tag{11.76b}$$

The controller canonical form realization of both $\mathbf{G}(D)$ and $\mathbf{G}'(D)$ has minimal overall constraint length $\nu = 2$; that is, it has $2^\nu = 4$ states.

Similarly, Example 11.7 shows that for a rate $R = 2/3$ code, a minimal encoder realization is obtained in observer canonical form. In general, for any rate $R = (n-1)/n$ encoder described by an $(n-1) \times n$ polynomial generator matrix $\mathbf{G}(D)$ with an equivalent systematic rational function generator matrix $\mathbf{G}'(D)$, the observer canonical form realization of $\mathbf{G}'(D)$ is minimal, provided all common

factors have been removed from the $1 \times n$ polynomial parity-check matrix $\mathbf{H}(D)$ corresponding to the systematic rational function parity-check matrix $\mathbf{H}'(D)$ derived from $\mathbf{G}'(D)$. (Removing common factors from a $1 \times n$ polynomial parity check matrix does not change its null space.) An exception occurs when one or more of the rational functions in $\mathbf{G}'(D)$ is not *realizable*; that is, the delay of its denominator polynomial exceeds the delay of its numerator polynomial, where the *delay* of a polynomial is defined as its lowest-degree nonzero term. In this case, a realizable minimal realization must be found, which may not correspond to observer canonical form. An example is given in Problem 11.9.

In general, the rate $R = k/n$ convolutional code produced by a $k \times n$ generator matrix $\mathbf{G}(D)$ can also be produced by any $k \times n$ matrix $\mathbf{G}'(D)$ with the same row space as $\mathbf{G}(D)$. If the encoder contains feedback, the code is recursive; otherwise, it is nonrecursive. Any realization of one of these equivalent generator matrices that has minimal overall constraint length, that is, a minimal number of delay elements, results in a minimal encoder. The foregoing summary of the requirements for determining minimal encoder realizations applies to all code rates normally used in practice, that is, rates $R = 1/n$ and $R = (n-1)/n$. For code rates with $k > 1$ and $(n-k) > 1$, such as rates 2/5, 2/4, and 3/5, determining minimal encoder realizations is more complex. In these cases, a minimal encoder may require a realization that is neither in controller canonical form nor in observer canonical form. The subject of minimal encoder realizations was first treated in a seminal paper by Forney [13]. More recently, the book by Johannesson and Zigangirov [12] provides a comprehensive treatment of this topic.

Throughout the remainder of the text, when we refer to an (n, k, ν) convolutional code, we assume a minimal encoder realization; that is, ν is the minimal overall constraint length of any encoder realization. Also, when we use the term *systematic* (or *nonsystematic*) code, we mean the code produced by a systematic (or nonsystematic) encoder. Finally, we observe that for rate $R = 1/n$ encoders in controller canonical form and rate $R = (n-1)/n$ encoders in observer canonical form, the overall constraint length ν equals the memory order m. Thus, for rates $R = 1/n$ and $R = (n-1)/n$, we can refer to an (n, k, ν) code or, equivalently, an (n, k, m) code. In general, however, if the form of the encoder realization is not specified, $m \leq \nu$.

To this point we have discussed nonsystematic feedforward encoders, systematic feedforward encoders, and systematic feedback encoders. A fourth general type of encoder is the class of *nonsystematic feedback encoders*.

EXAMPLE 11.8 A Rate $R = 2/3$ Nonsystematic Feedback Convolutional Encoder

Consider the rate $R = 2/3$ *nonsystematic recursive convolutional code* (NSRCC) generated by the matrix

$$\mathbf{G}(D) = \begin{bmatrix} 1/(1+D+D^2) & D/(1+D^3) & 1/(1+D^3) \\ D^2/(1+D^3) & 1/(1+D^3) & 1/(1+D) \end{bmatrix} \tag{11.77}$$

in nonsystematic feedback form. To determine the controller (observer) canonical form realization of (11.77), the rational functions in each row (column) of $\mathbf{G}(D)$ must

have the same denominator (feedback) polynomial; that is, we must find the least common denominator for the rational functions in each row (column). In this case, since $1/(1+D+D^2) = (1+D)/(1+D^3)$, and $1/(1+D) = (1+D+D^2)/(1+D^3)$, we can rewrite $\mathbf{G}(D)$ as

$$\mathbf{G}'(D) = \begin{bmatrix} (1+D)/(1+D^3) & D/(1+D^3) & 1/(1+D^3) \\ D^2/(1+D^3) & 1/(1+D^3) & (1+D+D^2)/(1+D^3) \end{bmatrix}. \quad (11.78)$$

We can now determine controller and observer canonical form realizations of the rate $R = 2/3$ NSRCC generated by $\mathbf{G}(D)$ directly from $\mathbf{G}'(D)$. These realizations are shown in Figure 11.9. Note that the controller canonical form results in a 64-state ($\nu = 6$) realization, whereas the observer canonical form results in a 512-state ($\nu = 9$) realization. In Problem 11.10 it is shown that a minimal 8-state ($\nu = 3$) observer canonical form realization is achieved by converting $\mathbf{G}(D)$ to systematic feedback form, thus resulting in a (3, 2, 3) SRCC.

Example 11.8 and Problem 11.10 illustrate that the same approach used to find minimal realizations for nonsystematic feedforward encoders can also be used for nonsystematic feedback encoders. To summarize our discussion of encoder realizations for convolutional codes, we show in Figure 11.10 the controller and observer canonical form realizations of an arbitrary realizable rational function

$$\mathbf{g}(D) = \mathbf{f}(D)/\mathbf{q}(D) = (f_0 + f_1 D + \ldots + f_m D^m)/(1 + q_1 D + \ldots + q_m D^m), \quad (11.79)$$

where $\mathbf{q}(D)$ is required to have a nonzero constant term to guarantee realizability.

We are now in a position to specify the notation that we will use to describe convolutional codes throughout the remainder of the text for the most practically important code rates, that is, for rates $R = 1/n$ and $R = (n-1)/n$. Any rate $R = 1/n$ convolutional code can be described (see Example 11.6) by a minimal nonsystematic feedforward encoder with polynomial generator matrix

$$\mathbf{G}(D) = [\mathbf{g}^{(0)}(D) \quad \mathbf{g}^{(1)}(D) \quad \cdots \quad \mathbf{g}^{(n-1)}(D)] \quad (11.80a)$$

or its rational function systematic feedback equivalent,

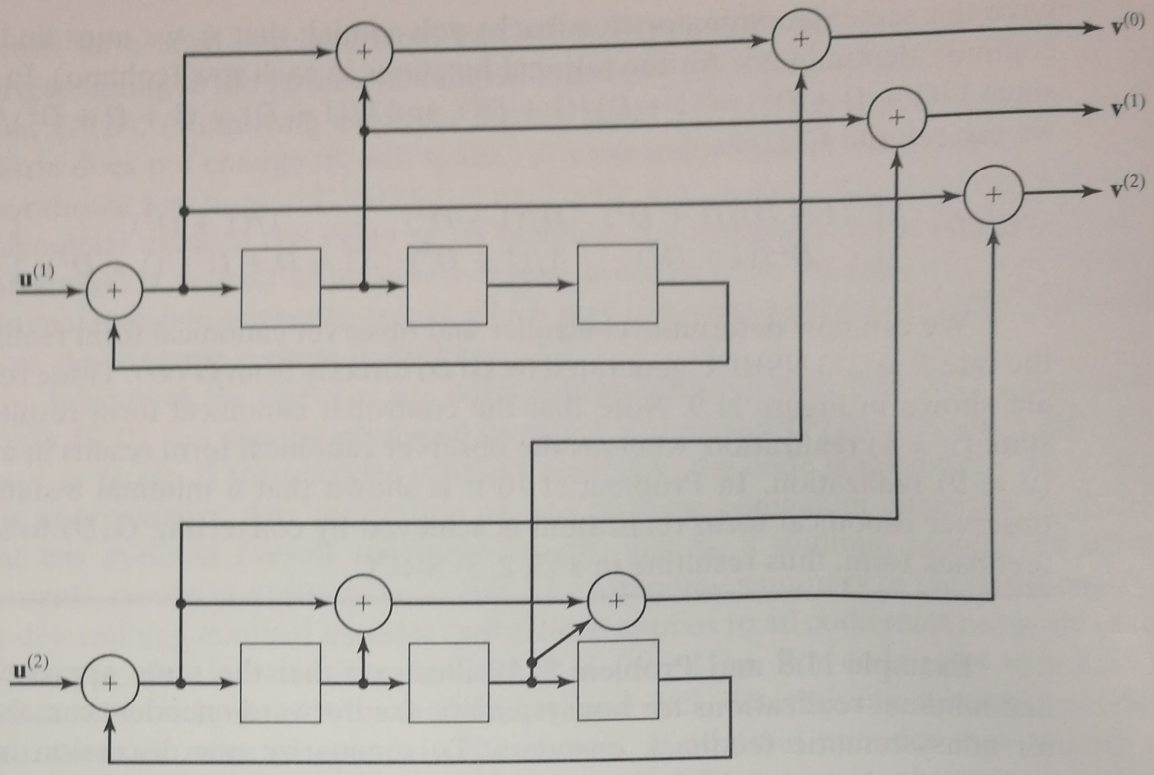
$$\mathbf{G}'(D) = [1 \quad \mathbf{g}^{(1)}(D)/\mathbf{g}^{(0)}(D) \quad \cdots \quad \mathbf{g}^{(n-1)}(D)/\mathbf{g}^{(0)}(D)]. \quad (11.80b)$$

(If $\mathbf{g}^{(0)}(D) = 1$, (11.80) describes a systematic feedforward encoder.) The code is described by specifying the n generator polynomials $\mathbf{g}^{(0)}(D), \mathbf{g}^{(1)}(D), \cdots, \mathbf{g}^{(n-1)}(D)$ or the n equivalent generator sequences $\mathbf{g}^{(0)}, \mathbf{g}^{(1)}, \cdots, \mathbf{g}^{(n-1)}$. Similarly, any rate $R = (n-1)/n$ convolutional code can be described (see Example 11.7) by a minimal systematic feedback encoder with rational function parity-check matrix

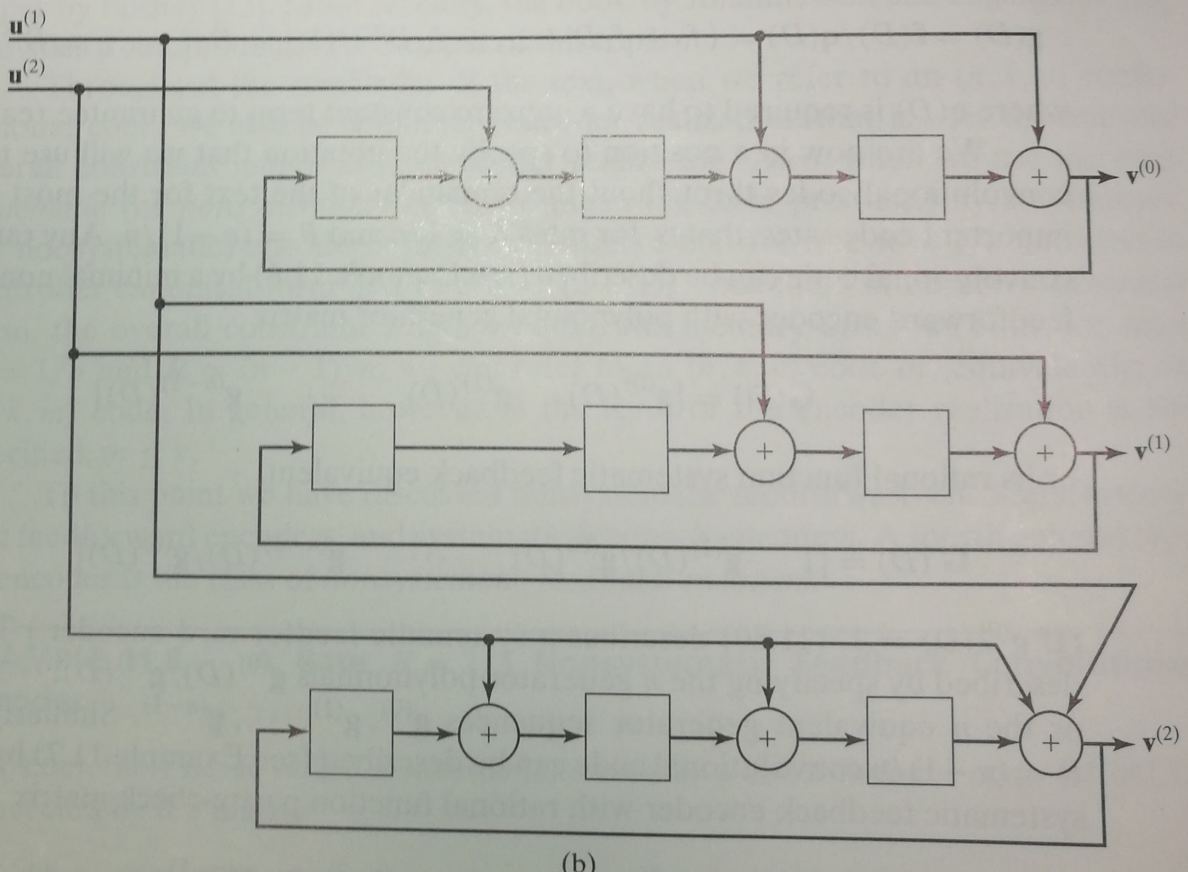
$$\mathbf{H}'(D) = [\mathbf{h}^{(0)}(D)/\mathbf{h}^{(n-1)}(D) \quad \cdots \quad \mathbf{h}^{(n-2)}(D)/\mathbf{h}^{(n-1)}(D) \quad 1] \quad (11.81a)$$

or its polynomial nonsystematic feedforward equivalent,

$$\mathbf{H}(D) = [\mathbf{h}^{(0)}(D) \quad \cdots \quad \mathbf{h}^{(n-2)}(D) \quad \mathbf{h}^{(n-1)}(D)]. \quad (11.81b)$$



(a)



(b)

FIGURE 11.9: (a) The controller canonical form and (b) the observer canonical form realization of a nonsystematic feedback encoder.

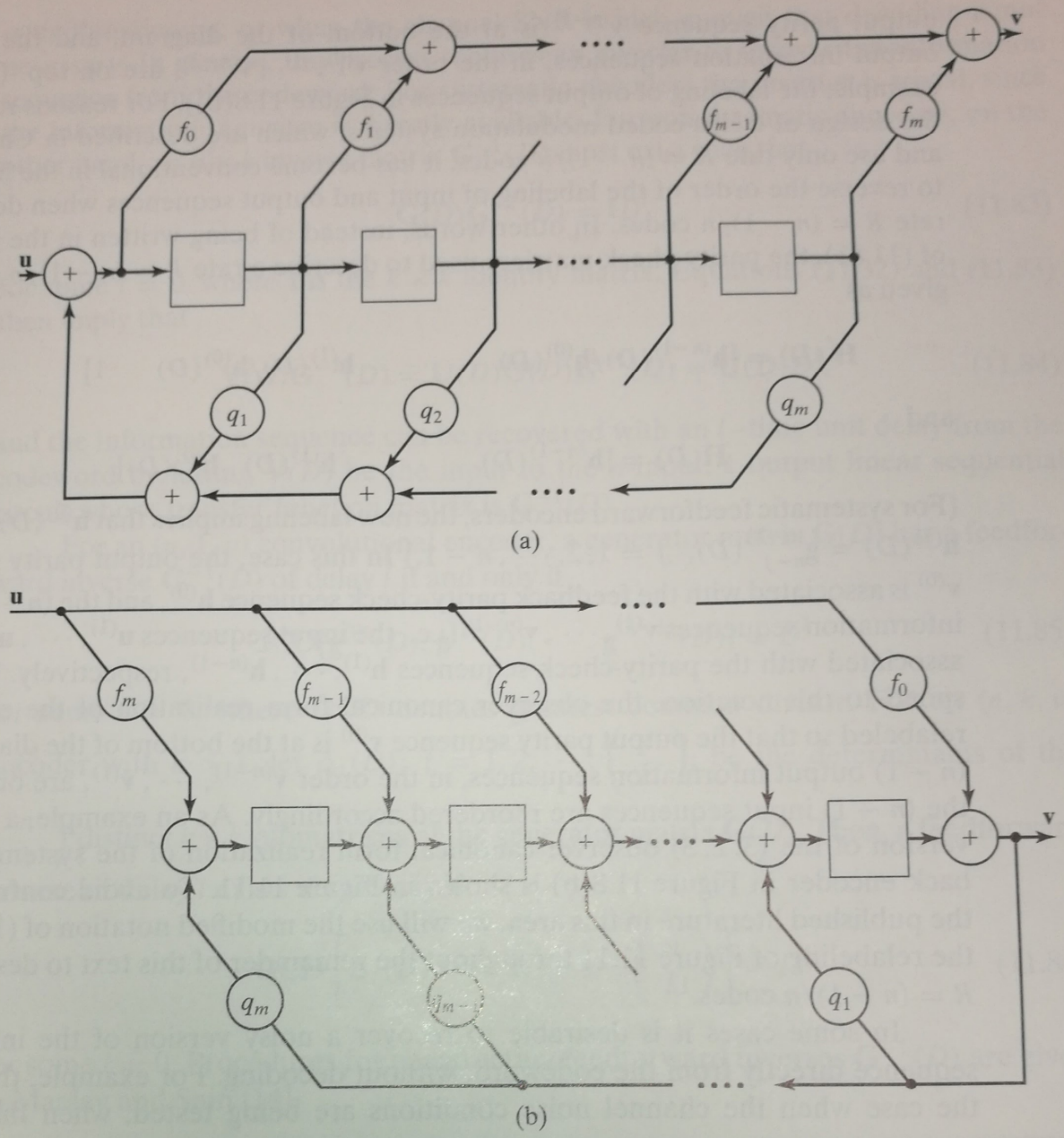


FIGURE 11.10: (a) The controller canonical form and (b) the observer canonical form realization of an arbitrary realizable rational function.

(If $\mathbf{h}^{(n-1)}(D) = 1$, (11.81) describes a systematic feedforward encoder.) The code is described by specifying the n parity-check polynomials $\mathbf{h}^{(0)}(D), \mathbf{h}^{(1)}(D), \cdots, \mathbf{h}^{(n-1)}(D)$ or the n equivalent parity-check sequences $\mathbf{h}^{(0)}, \mathbf{h}^{(1)}, \cdots, \mathbf{h}^{(n-1)}$.

In the notation of (11.81), for rate $R = (n-1)/n$ codes, the output parity sequence $\mathbf{v}^{(n-1)}$ is associated with the feedback parity-check sequence $\mathbf{h}^{(n-1)}$, and the $(n-1)$ output information sequences $\mathbf{v}^{(0)}, \cdots, \mathbf{v}^{(n-2)}$ (i.e., the input sequences $\mathbf{u}^{(1)}, \cdots, \mathbf{u}^{(n-1)}$) are associated with the parity-check sequences $\mathbf{h}^{(0)}, \cdots, \mathbf{h}^{(n-2)}$, respectively. Also, in the observer canonical form realization of the encoder, the

output parity sequence $\mathbf{v}^{(n-1)}$ is at the bottom of the diagram, and the $(n-1)$ output information sequences, in the order $\mathbf{v}^{(0)}, \cdots, \mathbf{v}^{(n-2)}$, are on top. (See, for example, the labeling of output sequences in Figure 11.8(b)). For reasons related to the design of trellis-coded modulation systems, which are described in Chapter 18 and use only rate $R = (n-1)/n$ codes, it has become conventional in the literature to reverse the order of the labeling of input and output sequences when describing rate $R = (n-1)/n$ codes. In other words, instead of being written in the notation of (11.81), the parity-check matrices used to describe a rate $R = (n-1)/n$ code are given as

$$\mathbf{H}'(D) = [\mathbf{h}^{(n-1)}(D)/\mathbf{h}^{(0)}(D) \quad \cdots \quad \mathbf{h}^{(1)}(D)/\mathbf{h}^{(0)}(D) \quad 1] \tag{11.82a}$$

and

$$\mathbf{H}(D) = [\mathbf{h}^{(n-1)}(D) \quad \cdots \quad \mathbf{h}^{(1)}(D) \quad \mathbf{h}^{(0)}(D)]. \tag{11.82b}$$

(For systematic feedforward encoders, the new labeling implies that $\mathbf{h}^{(0)}(D) = 1$, and $\mathbf{h}^{(j)}(D) = \mathbf{g}_{n-j}^{(n-1)}(D)$, $j = 1, 2, \cdots, n-1$.) In this case, the output parity sequence $\mathbf{v}^{(0)}$ is associated with the feedback parity-check sequence $\mathbf{h}^{(0)}$, and the $(n-1)$ output information sequences $\mathbf{v}^{(1)}, \cdots, \mathbf{v}^{(n-1)}$ (i.e., the input sequences $\mathbf{u}^{(1)}, \cdots, \mathbf{u}^{(n-1)}$) are associated with the parity-check sequences $\mathbf{h}^{(1)}, \cdots, \mathbf{h}^{(n-1)}$, respectively. To correspond to this notation, the observer canonical form realization of the encoder is relabeled so that the output parity sequence $\mathbf{v}^{(0)}$ is at the bottom of the diagram, the $(n-1)$ output information sequences, in the order $\mathbf{v}^{(n-1)}, \cdots, \mathbf{v}^{(1)}$, are on top, and the $(n-1)$ input sequences are reordered accordingly. As an example, a relabeled version of the $(3, 2, 3)$ observer canonical form realization of the systematic feedback encoder in Figure 11.8(b) is shown in Figure 11.11. To avoid confusion with the published literature in this area, we will use the modified notation of (11.82) and the relabeling of Figure 11.11 throughout the remainder of this text to describe rate $R = (n-1)/n$ codes.[1]

In some cases it is desirable to recover a noisy version of the information sequence directly from the codeword, without decoding. For example, this may be the case when the channel noise conditions are being tested, when the decoder

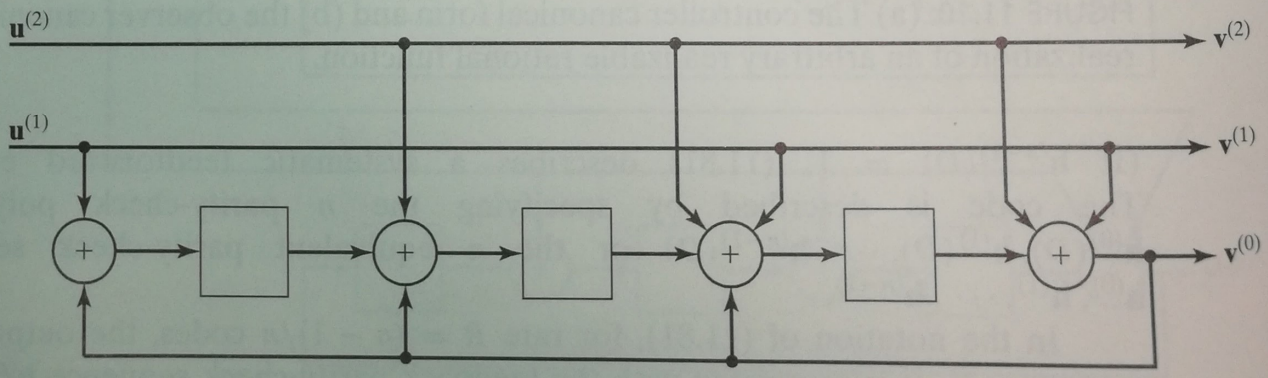


FIGURE 11.11: A relabeled version of the $(3, 2, 3)$ systematic feedback encoder of Figure 11.8(b).

[1] Exceptions are made in Sections 13.5–13.7 and in Chapter 21 where, to conform to the literature on systematic feedforward encoders, the labeling of Figure 11.5 is used to describe rate $R = (n-1)/n$ codes.

is malfunctioning, or when the channel SNR is high enough that decoding is not necessary. In general, this process requires an inverter to recover the information sequence from the codeword. For systematic encoders the inverter is trivial, since the information sequence is directly available. For nonsystematic encoders, on the other hand, an $n \times k$ inverse matrix $\mathbf{G}^{-1}(D)$ must exist such that

$$\mathbf{G}(D)\mathbf{G}^{-1}(D) = \mathbf{I}D^l \tag{11.83}$$

for some $l \geq 0$, where $\mathbf{I}$ is the $k \times k$ identity matrix. Equations (11.32) and (11.83) then imply that

$$\mathbf{V}(D)\mathbf{G}^{-1}(D) = \mathbf{U}(D)\mathbf{G}(D)\mathbf{G}^{-1}(D) = \mathbf{U}(D)D^l, \tag{11.84}$$

and the information sequence can be recovered with an l–time unit delay from the codeword by letting $\mathbf{V}(D)$ be the input to the n-input, k-output linear sequential circuit whose transfer function matrix is $\mathbf{G}^{-1}(D)$.

For an $(n, 1, \nu)$ convolutional encoder, a generator matrix $\mathbf{G}(D)$ has a feedforward inverse $\mathbf{G}^{-1}(D)$ of delay l if and only if

$$\text{GCD}[\mathbf{g}^{(0)}(D), \mathbf{g}^{(1)}(D), \cdots, \mathbf{g}^{(n-1)}(D)] = D^l \tag{11.85}$$

for some $l \geq 0$, where GCD denotes greatest common divisor. For an (n, k, ν) encoder with $k > 1$, let $\Delta_i(D)$, $i = 1, 2, \cdots, \binom{n}{k}$, be the determinants of the $\binom{n}{k}$ distinct $k \times k$ submatrices of the generator matrix $\mathbf{G}(D)$. Then, a feedforward inverse of delay l exists if and only if

$$\text{GCD}\left[\Delta_i(D) : i = 1, 2, \cdots, \binom{n}{k}\right] = D^l \tag{11.86}$$

for some $l \geq 0$. Procedures for constructing feedforward inverses $\mathbf{G}^{-1}(D)$ are given in Massey and Sain [14].

EXAMPLE 11.9 Feedforward Encoder Inverses

a. For the (2, 1, 3) encoder of Figure 11.1, the generator matrix is given by

$$\mathbf{G}(D) = \begin{bmatrix} \mathbf{g}^{(0)}(D) & \mathbf{g}^{(1)}(D) \end{bmatrix} = \begin{bmatrix} 1 + D^2 + D^3 & 1 + D + D^2 + D^3 \end{bmatrix}, \tag{11.87a}$$

$$\text{GCD}[\mathbf{g}^{(0)}(D), \ \mathbf{g}^{(1)}(D)] = \text{GCD}[1+D^2+D^3, \ 1+D+D^2+D^3] = 1, \tag{11.87b}$$

and the transfer function matrix

$$\mathbf{G}^{-1}(D) = \begin{bmatrix} 1 + D + D^2 \\ D + D^2 \end{bmatrix} \tag{11.88}$$

provides the required feedforward inverse of delay 0; that is, $\mathbf{G}(D)\mathbf{G}^{-1}(D) = 1$. A realization of the encoder inverse is shown in Figure 11.12(a).

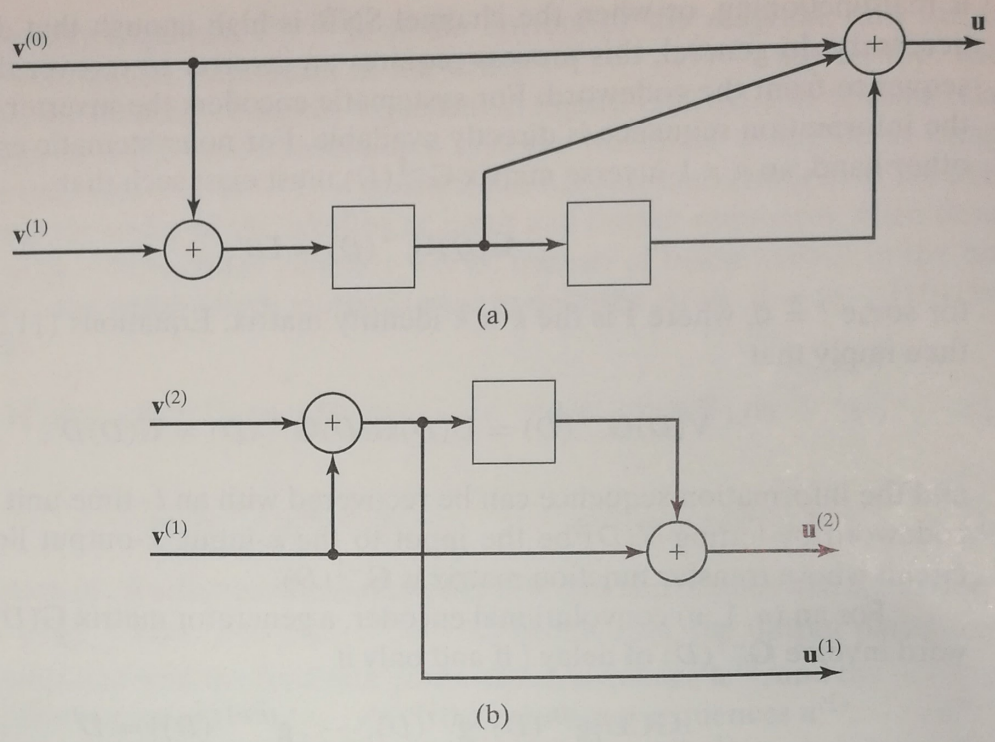


FIGURE 11.12: Feedforward encoder inverses for (a) a (3, 1, 2) encoder and (b) a (3, 2, 2) encoder.

b. For the (3, 2, 2) encoder of Figure 11.2 with generator matrix $\mathbf{G}(D)$ given in (11.34), the 2×2 submatrices of $\mathbf{G}(D)$ yield determinants $1 + D + D^2$, $1 + D^2$, and 1. Because

$$\text{GCD}[1 + D + D^2,\ 1 + D^2,\ 1] = 1, \tag{11.89}$$

there exists a feedforward inverse of delay 0. The required transfer function matrix is given by

$$\mathbf{G}^{-1}(D) = \begin{bmatrix} 0 & 0 \\ 1 & 1+D \\ 1 & D \end{bmatrix}, \tag{11.90}$$

and its realization is shown in Figure 11.12(b).

To understand what happens when a feedforward inverse does not exist, it is best to consider another example.

EXAMPLE 11.10 A Catastrophic Encoder

Consider the (2, 1, 2) encoder whose generator matrix is given by

$$\mathbf{G}(D) = \left[\mathbf{g}^{(0)}(D)\ \ \mathbf{g}^{(1)}(D)\right] = \left[1 + D \quad 1 + D^2\right]. \tag{11.91a}$$

In this case we see that

$$\text{GCD}[\mathbf{g}^{(0)}(D)\ \ \mathbf{g}^{(1)}(D)] = \text{GCD}[1 + D,\ \ 1 + D^2] = 1 + D, \tag{11.91b}$$

and a feedforward inverse does not exist. If the information sequence is $\mathbf{u}(D) = \frac{1}{1+D} = 1 + D + D^2 + \cdots$, then the output sequences are $\mathbf{v}^{(0)}(D) = 1$ and $\mathbf{v}^{(1)}(D) = 1 + D$; that is, the codeword has only weight 3 even though the information sequence has infinite weight. If this codeword is transmitted over a BSC, and the three nonzero bits are changed to zeros by the channel noise, then the received sequence will be all zero. A maximum likelihood decoder will then produce the all-zero codeword as its estimate, since this is a valid codeword and it agrees exactly with the received sequence. Thus, the estimated information sequence will be $\mathbf{u}(D) = \mathbf{0}(D)$, implying an infinite number of decoding errors caused by a finite number (only three in this case) of channel errors. Clearly, this is very undesirable, and the encoder is said to be subject to catastrophic error propagation; that is, it is a *catastrophic encoder*.

Thus, in selecting encoders for use in an error control coding system, it is important to avoid the selection of catastrophic encoders. Massey and Sain [14] have shown that (11.85) and (11.86) are necessary and sufficient conditions for an encoder to be *noncatastrophic*. Hence, any encoder for which a feedforward inverse exists is noncatastrophic. It follows that systematic encoders are always noncatastrophic, since a trivial feedforward inverse exists. Minimal nonsystematic encoders are also noncatastrophic, but nonminimal encoders can be catastrophic [12]. Note, for example, that the nonminimal encoder of (11.75) does not satisfy (11.85), and thus it is catastrophic. Only noncatastrophic encoders are listed in the tables of good convolutional encoders given later in the text in Chapters 12, 13, 16, 18, and 21.

An (n, k, ν) convolutional encoder can generate infinitely long encoded sequences (codewords). Because n encoded bits are produced for each k information bits, $R = k/n$ is called the rate of the convolutional encoder. For a finite-length information sequence of h time units, or $K^* = kh$ bits, it is customary to begin encoding in the all-zero state; that is, the initial contents of the shift registers are all zeros. Also, to facilitate decoding, it is necessary to return the encoder to the all-zero state after the last information block has entered the encoder, as will be seen in Chapters 12 and 13. This requires that an additional m input blocks enter the encoder to force it to return to the all-zero state. For feedforward encoders, it can be seen from the encoder diagram that these m termination blocks must be all zeros, whereas for feedback encoders, the termination blocks depend on the information sequence (see Problem 11.12). These termination blocks are not part of the information sequence, since they are fixed and cannot be chosen arbitrarily. (We note here that for controller canonical form encoder realizations with $\nu < km$, only ν bits are required for termination. Thus, $km - \nu$ bits in the m termination blocks can be information bits. For example, for the (4, 3, 3) encoder realization shown in Figure 11.3, only $\nu = 3$ bits are needed for termination, and the remaining 3 bits in the 2 termination blocks can be information bits. A similar situation holds in the case of observer canonical form realizations (see Problem 11.13). For simplicity; however, we ignore these special cases in the succeeding development.) The corresponding codeword has length $N = n(h + m)$, including the m output blocks, or nm output bits, resulting from termination. Such a terminated convolutional code can be viewed as an $(N, K^*) = [n(h + m),\ kh]$ linear block code with a $K^* \times N$ generator matrix $\mathbf{G}$. The block (terminated convolutional) code rate is given by

$R_t = \frac{K^*}{N} = \frac{kh}{n(h+m)}$, the ratio of the number of information bits to the length of the codeword. If $h >> m$, then $h/(h+m) \approx 1$, and the block (terminated convolutional) code rate R_t and the convolutional encoder rate R are approximately equal. This represents the normal mode of operation for convolutional codes, and we usually do not distinguish between the rate R of a convolutional encoder and the rate R_t of the associated block (terminated convolutional) code; that is, we assume $R = R_t$. If h is small, however, the ratio $\frac{kh}{n(h+m)}$, which is the rate R_t of the block (terminated convolutional) code, is reduced below the convolutional encoder rate R by a fractional amount

$$\frac{\frac{k}{n} - \frac{kh}{n(h+m)}}{k/n} = \frac{m}{h+m} \tag{11.92}$$

called the *fractional rate loss*. For example, for the (2, 1, 3) encoder and information sequence in Example 11.1, $h = 5$ and the fractional rate loss is $\frac{3}{8} = 37.5\%$; however, for an information sequence of length $h = 1000$, the fractional rate loss is only $\frac{3}{1003} = 0.3\%$.

For short information sequences, the fractional rate loss can be eliminated by simply not transmitting the nm termination bits and considering the first $N^* = nh$ encoder output bits as the codeword; however, as will be seen in Chapters 12 and 13, if the termination bits are not transmitted, the performance of the code is degraded. One way to combat this problem is to allow the encoder to start in an arbitrary state—that is, the initial contents of the shift registers are not necessarily all zeros—and then force it to return to the same state at the end of the information sequence. It turns out that if the starting state is chosen properly, no extra bits are required to force the encoder back to the same state, so no fractional rate loss is incurred. This technique results in so-called tail-biting codes, a subject that is discussed in more detail in Section 12.7.

11.2 STRUCTURAL PROPERTIES OF CONVOLUTIONAL CODES

Because a convolutional encoder is a linear sequential circuit, its operation can be described by a state diagram. The state of an encoder is defined as its shift register contents. For an (n, k, ν) encoder in controller canonical form, the ith shift register at time unit l (when $u_l^{(1)}, u_l^{(2)}, \cdots, u_l^{(k)}$ are the encoder inputs) contains ν_i bits, denoted as $s_{l-1}^{(i)}, s_{l-2}^{(i)}, \cdots, s_{l-\nu_i}^{(i)}$, where $s_{l-1}^{(i)}$ represent the contents of the leftmost delay element, and $s_{l-\nu_i}^{(i)}$ represents the contents of the rightmost delay element, $1 \le i \le k$.

Definition 11.6 The *encoder state* σ_l at time unit l is the binary ν-tuple

$$\sigma_l = (s_{l-1}^{(1)} s_{l-2}^{(1)} \cdots s_{l-\nu_1}^{(1)} s_{l-1}^{(2)} s_{l-2}^{(2)} \cdots s_{l-\nu_2}^{(2)} \cdots s_{l-1}^{(k)} s_{l-2}^{(k)} \cdots s_{l-\nu_k}^{(k)}). \tag{11.93}$$

Thus, we see from (11.20) that there are a total of 2^ν different possible states. For an $(n, 1, \nu)$ encoder, the encoder state at time unit l is simply

$$\sigma_l = (s_{l-1}\ s_{l-2}\ \cdots\ s_{l-\nu}). \tag{11.94}$$

In the case of feedforward encoders, we can see from the encoder diagram (see, for example, Figure 11.1) that the contents of the ith shift register at time unit

l are its previous ν_i inputs, that is, $u_{l-1}^{(i)}, u_{l-2}^{(i)}, \cdots, u_{l-\nu_i}^{(i)}$, and hence the encoder state at time unit l is given by

$$\sigma_k = (u_{l-1}^{(1)} u_{l-2}^{(1)} \cdots u_{l-\nu_1}^{(1)} u_{l-1}^{(2)} u_{l-2}^{(2)} \cdots u_{l-\nu_2}^{(2)} \cdots u_{l-1}^{(k)} u_{l-2}^{(k)} \cdots u_{l-\nu_k}^{(k)}). \tag{11.95}$$

For an $(n, 1, \nu)$ encoder, (11.93) becomes simply

$$\sigma = (u_{l-1} \; u_{l-2} \; \cdots \; u_{l-\nu}). \tag{11.96}$$

For observer canonical form encoder realizations, the encoder state σ_l at time unit l can also be defined as in (11.93), except that in this case there are, in general, a total of n rather than k shift registers. Also, feedforward encoders in observer canonical form do not have the property that the contents of a shift register are its ν_i previous inputs; that is, (11.95) and (11.96) apply only to feedforward encoder realizations in controller canonical form.

Each new input block of k bits causes the encoder to shift; that is, there is a transition to a new state. Hence, there are 2^k branches leaving each state in the state diagram, one corresponding to each different input block. For an $(n, 1, \nu)$ encoder, therefore, there are only two branches leaving each state. Each branch in the state diagram is labeled with the k inputs $(u_l^{(1)}, u_l^{(2)}, \cdots, u_l^{(k)})$ causing the transition and the n corresponding outputs $(v_l^{(0)}, v_l^{(1)}, \cdots, v_l^{(n-1)})$. The state diagrams for the feedforward controller canonical form realizations of the (2, 1, 3) encoder of Figure 11.1 and the (3, 2, 2) encoder of Figure 11.2 are shown in Figures 11.13(a) and 11.13(b), respectively. The states are labeled $S_0, S_1, \cdots, S_{2^\nu - 1}$, where by convention S_i represents the state whose binary ν-tuple representation $b_0, b_1, \cdots, b_{\nu-1}$ is equivalent to the integer $i = b_0 2^0 + b_1 2^1 + \cdots + b_{\nu-1} 2^{\nu-1}$. For example, for the (2, 1, 3) encoder of Figure 11.1, if the shift register contents at time l are given by $\sigma_l = (s_{l-1} \; s_{l-2} \; s_{l-3}) = (u_{l-1} \; u_{l-2} \; u_{l-3})$, the state at time l is denoted as S_i, where $i = u_{l-1} + 2u_{l-2} + 4u_{l-3}$.

We note here that for *time-invariant convolutional encoders*, that is, encoders whose generator matrix and associated encoder diagram do not change with time, the encoder state diagram is also time-invariant. Thus the same state diagram (see, e.g., those in Figure 11.13) is valid for all time units l. *Time-varying convolutional encoders* are not treated explicitly in this text, but we note here that the class of turbo codes, to be covered in Chapter 16, is generated by a special kind of time-varying convolutional encoder.

Assuming that the encoder is initially in state S_0 (the all-zero state), we can obtain the codeword corresponding to any given information sequence by following the path through the state diagram determined by the information sequence and noting the corresponding outputs on the branch labels. Following the last information block, a terminated encoder is returned to state S_0 by a sequence of m input blocks appended to the information sequence. For example, in Figure 11.13(a), if $\mathbf{u} = (1\ 1\ 1\ 0\ 1)$, encoding is terminated by appending $m = 3$ zeros to $\mathbf{u}$, and the resulting codeword is given by $\mathbf{v} = (1\,1, 1\,0, 0\,1, 0\,1, 1\,1, 1\,0, 1\,1, 1\,1)$.

We now return briefly to the subject of catastrophic encoders introduced in the previous section. An encoder is catastrophic if and only if the state diagram contains a cycle with zero output weight other than the zero-weight cycle around the state S_0. The state diagram for the catastrophic encoder of Example 11.10 is

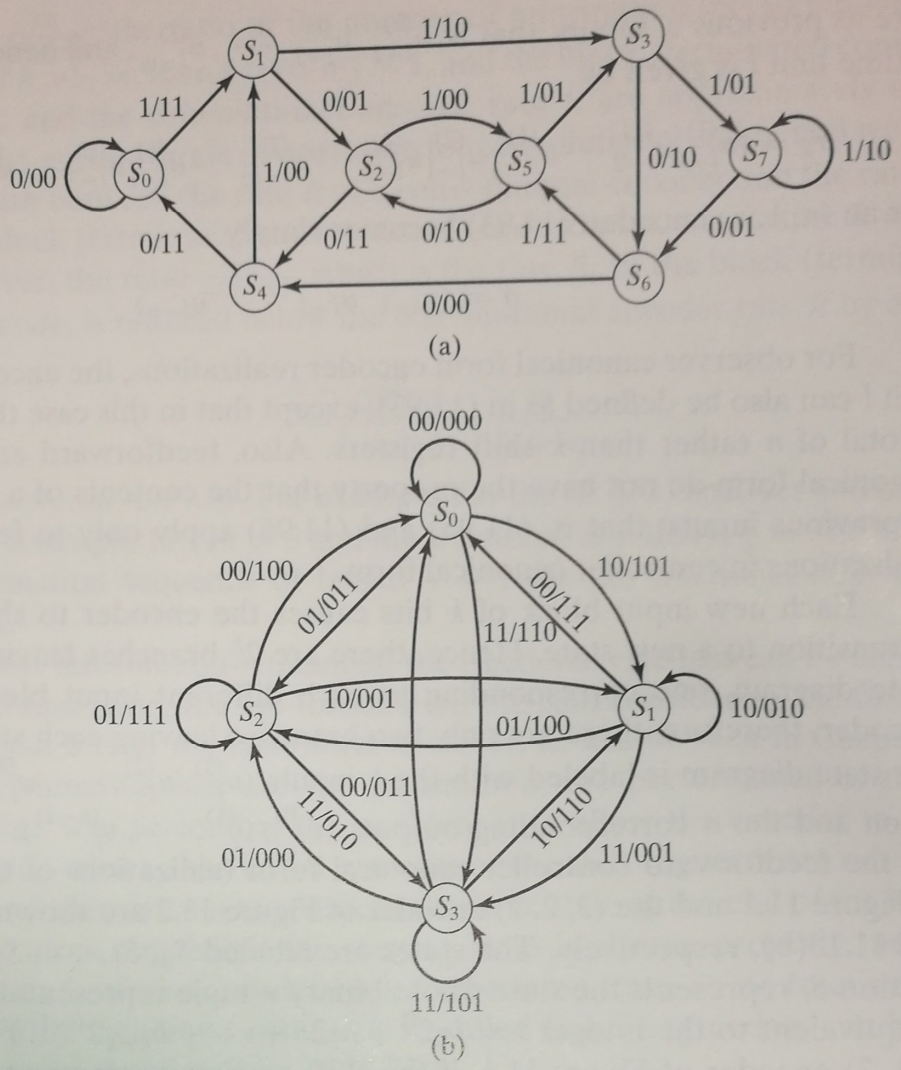


FIGURE 11.13: Encoder state diagrams for (a) the (2, 1, 3) encoder of Figure 11.1 and (b) the (3, 2, 2) encoder of Figure 11.2.

shown in Figure 11.14. Note that the single branch cycle around the state S_3 has zero output weight.

The state diagram can be modified to provide a complete description of the Hamming weights of all nonzero *codewords*, that is, a *codeword weight enumerating function* (WEF) for the code. We begin by considering the case of unterminated encoders; that is, the information sequence can be of arbitrary length. State S_0 is split into an initial state and a final state, the zero-weight branch around state S_0 is deleted, and each branch is labeled with a *branch gain* X^d, where d is the weight of the n encoded bits on that branch. Each path connecting the initial state to the final state represents a nonzero codeword that diverges from and remerges with state S_0 exactly once. (We do not enumerate codewords that never remerge with state S_0, since, for noncatastrophic encoders, they must have infinite weight. Also, codewords that diverge from and remerge with state S_0 more than once are not enumerated, since they can be considered as a sequence of shorter codewords.) The *path gain* is the product of the branch gains along a path, and the weight of the associated codeword is the power of X in the path gain. The modified state diagrams for the encoders of Figures 11.1 and 11.2 are shown in Figures 11.15(a) and 11.15(b), respectively.

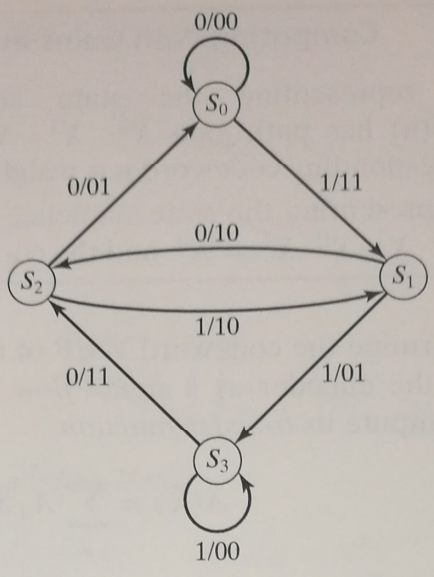
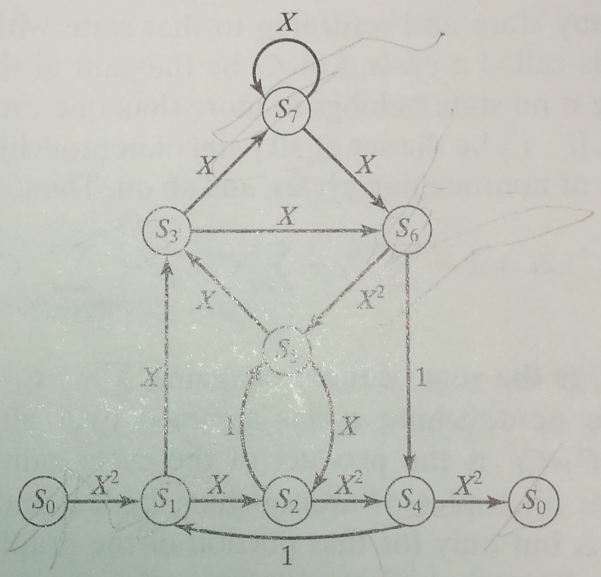
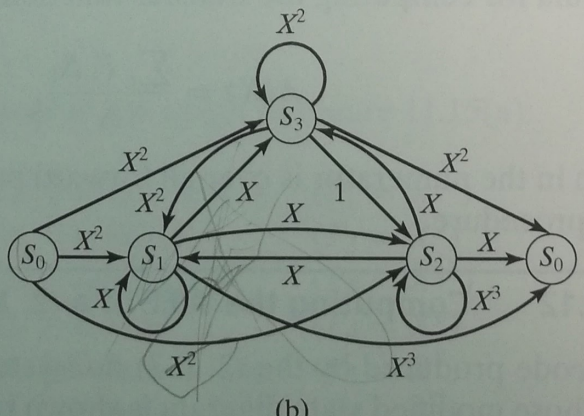


FIGURE 11.14: State diagram of a (2, 1, 2) catastrophic encoder.



(a)



(b)

FIGURE 11.15: Modified encoder state diagrams for (a) the (2, 1, 3) encoder of Figure 11.1 and (b) the (3, 2, 2) encoder of Figure 11.2.

EXAMPLE 11.11 Computing Path Gains in a State Diagram

a. The path representing the state sequence $S_0S_1S_3S_7S_6S_5S_2S_4S_0$ in Figure 11.15(a) has path gain $X^2 \cdot X^1 \cdot X^1 \cdot X^1 \cdot X^2 \cdot X^1 \cdot X^2 \cdot X^2 = X^{12}$, and the corresponding codeword has weight 12.

b. The path representing the state sequence $S_0S_1S_3S_2S_0$ in Figure 11.15(b) has path gain $X^2 \cdot X^1 \cdot X^0 \cdot X^1 = X^4$, and the corresponding codeword has weight 4.

We can determine the codeword WEF of a code by considering the modified state diagram of the encoder as a *signal flow graph* and applying Mason's gain formula [15] to compute its *transfer function*,

$$A(X) = \sum_d A_d X^d, \tag{11.97}$$

where A_d is the number of codewords of weight d. In a signal flow graph, a path connecting the initial state to the final state that does not go through any state twice is called a *forward path*. Let F_i be the gain of the ith forward path. A closed path starting at any state and returning to that state without going through any other state twice is called a *cycle*. Let C_i be the gain of the ith cycle. A set of cycles is *nontouching* if no state belongs to more than one cycle in the set. Let $\{i\}$ be the set of all cycles, $\{i', j'\}$ be the set of all pairs of nontouching cycles, $\{i'', j'', l''\}$ be the set of all triples of nontouching cycles, and so on. Then, we define

$$\Delta = 1 - \sum_i C_i + \sum_{i',j'} C_{i'}C_{j'} - \sum_{i'',j'',l''} C_{i''}C_{j''}C_{l''} + \cdots, \tag{11.98}$$

where $\sum_i C_i$ is the sum of the cycle gains, $\sum_{i',j'} C_{i'}C_{j'}$ is the product of the cycle gains of two nontouching cycles summed over all pairs of nontouching cycles, $\sum_{i'',j'',l''} C_{i''}C_{j''}C_{l''}$ is the product of the cycle gains of three nontouching cycles summed over all triples of nontouching cycles, and so on. Finally, Δ_i is defined exactly like Δ but only for that portion of the graph not touching the ith forward path; that is, all states along the ith forward path, along with all branches connected to those states, are removed from the graph when computing Δ_i. We can now state Mason's formula for computing the transfer function $A(X)$ of a graph as

$$A(X) = \frac{\sum_i F_i \Delta_i}{\Delta}, \tag{11.99}$$

where the sum in the numerator is over all forward paths. We give several examples to clarify this procedure.

EXAMPLE 11.12 Computing the WEF of a (2, 1, 3) Code

Consider the code produced by the (2, 1, 3) nonsystematic feedforward encoder of Figure 11.1, whose modified state diagram is shown in Figure 11.15(a). There are 11 cycles in the graph of Figure 11.15(a):

Cycle 1:	$S_1S_3S_7S_6S_5S_2S_4S_1$	$(C_1 = X^8)$
Cycle 2:	$S_1S_3S_7S_6S_4S_1$	$(C_2 = X^3)$
Cycle 3:	$S_1S_3S_6S_5S_2S_4S_1$	$(C_3 = X^7)$
Cycle 4:	$S_1S_3S_6S_4S_1$	$(C_4 = X^2)$
Cycle 5:	$S_1S_2S_5S_3S_7S_6S_4S_1$	$(C_5 = X^4)$
Cycle 6:	$S_1S_2S_5S_3S_6S_4S_1$	$(C_6 = X^3)$
Cycle 7:	$S_1S_2S_4S_1$	$(C_7 = X^3)$
Cycle 8:	$S_2S_5S_2$	$(C_8 = X)$
Cycle 9:	$S_3S_7S_6S_5S_3$	$(C_9 = X^5)$
Cycle 10:	$S_3S_6S_5S_3$	$(C_{10} = X^4)$
Cycle 11:	S_7S_7	$(C_{11} = X)$

There are 10 pairs of nontouching cycles:

Cycle Pair 1:	(Cycle 2, Cycle 8)	$(C_2C_8 = X^4)$
Cycle Pair 2:	(Cycle 3, Cycle 11)	$(C_3C_{11} = X^8)$
Cycle Pair 3:	(Cycle 4, Cycle 8)	$(C_4C_8 = X^3)$
Cycle Pair 4:	(Cycle 4, Cycle 11)	$(C_4C_{11} = X^3)$
Cycle Pair 5:	(Cycle 6, Cycle 11)	$(C_6C_{11} = X^4)$
Cycle Pair 6:	(Cycle 7, Cycle 9)	$(C_7C_9 = X^8)$
Cycle Pair 7:	(Cycle 7, Cycle 10)	$(C_7C_{10} = X^7)$
Cycle Pair 8:	(Cycle 7, Cycle 11)	$(C_7C_{11} = X^4)$
Cycle Pair 9:	(Cycle 8, Cycle 11)	$(C_8C_{11} = X^2)$
Cycle Pair 10:	(Cycle 10, Cycle 11)	$(C_{10}C_{11} = X^5)$

There are 2 triples of nontouching cycles:

Cycle Triple 1: (Cycle 4, Cycle 8, Cycle 11)	$(C_4C_8C_{11} = X^4)$
Cycle Triple 2: (Cycle 7, Cycle 10, Cycle 11)	$(C_7C_{10}C_{11} = X^8)$

There are no other sets of nontouching cycles. Therefore,

$$\begin{aligned}\Delta &= 1 - (X^8 + X^3 + X^7 + X^2 + X^4 + X^3 + X^3 + X + X^5 + X^4 + X) \\ &\quad + (X^4 + X^8 + X^3 + X^3 + X^4 + X^8 + X^7 + X^4 + X^2 + X^5) - (X^4 + X^8) \\ &= 1 - 2X - X^3. \end{aligned} \tag{11.100}$$

There are 7 forward paths in the graph of Figure 11.15(a):

Forward Path 1:	$S_0S_1S_3S_7S_6S_5S_2S_4S_0$	$(F_1 = X^{12})$
Forward Path 2:	$S_0S_1S_3S_7S_6S_4S_0$	$(F_2 = X^7)$
Forward Path 3:	$S_0S_1S_3S_6S_5S_2S_4S_0$	$(F_3 = X^{11})$
Forward Path 4:	$S_0S_1S_3S_6S_4S_0$	$(F_4 = X^6)$
Forward Path 5:	$S_0S_1S_2S_5S_3S_7S_6S_4S_0$	$(F_5 = X^8)$
Forward Path 6:	$S_0S_1S_2S_5S_3S_6S_4S_0$	$(F_6 = X^7)$
Forward Path 7:	$S_0S_1S_2S_4S_0$	$(F_7 = X^7)$

Forward paths 1 and 5 touch all states in the graph, and hence the subgraph not touching these paths contains no states. Therefore,

$$\Delta_1 = \Delta_5 = 1. \tag{11.101}$$

The subgraph not touching forward paths 3 and 6 is shown in Figure 11.16(a), and hence,

$$\Delta_3 = \Delta_6 = 1 - X. \tag{11.102}$$

The subgraph not touching forward path 2 is shown in Figure 11.16(b), and hence,

$$\Delta_2 = 1 - X. \tag{11.103}$$

The subgraph not touching forward path 4 is shown in Figure 11.16(c), and hence,

$$\Delta_4 = 1 - (X + X) + (X^2) = 1 - 2X + X^2. \tag{11.104}$$

The subgraph not touching forward path 7 is shown in Figure 11.16(d), and hence,

$$\Delta_7 = 1 - (X + X^4 + X^5) + (X^5) = 1 - X - X^4. \tag{11.105}$$

The transfer function for this graph, that is, the WEF of the associated (2, 1, 3) code, is then given by

$$\begin{aligned} A(X) &= \frac{X^{12} \cdot 1 + X^7(1-X) + X^{11}(1-X) + X^6(1-2X+X^2) + X^8 \cdot 1 + X^7(1-X) + X^7(1-X-X^4)}{1-2X-X^3} \\ &= \frac{X^6 + X^7 - X^8}{1-2X-X^3} \\ &= X^6 + 3X^7 + 5X^8 + 11X^9 + 25X^{10} + \cdots . \end{aligned} \tag{11.106}$$

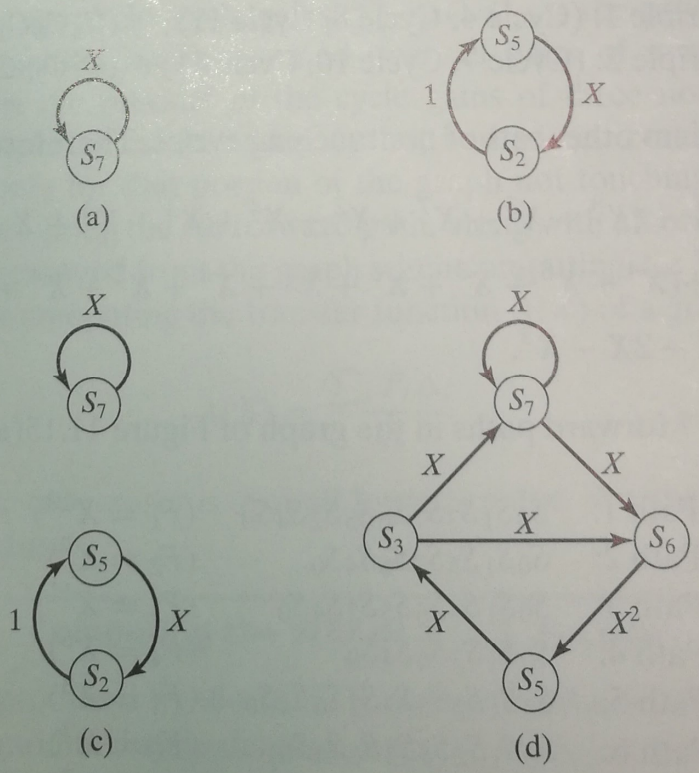


FIGURE 11.16: Subgraphs for computing Δ_i in Example 11.12.

The codeword WEF $A(X)$ provides a complete description of the weight distribution of all nonzero codewords that diverge from and remerge with state S_0 exactly once. In this case there is one such codeword of weight 6, three of weight 7, five of weight 8, and so on.

EXAMPLE 11.13 Computing the WEF of a (3, 2, 2) Code

Consider the code produced by the (3, 2, 2) nonsystematic feedforward encoder of Figure 11.2, whose modified state diagram is shown in Figure 11.15(b). It can be shown that (see Problem 11.17) there are 8 cycles, 6 pairs of nontouching cycles, and 1 triple of nontouching cycles in the graph of Figure 11.15(b), and

$$\begin{aligned}\Delta &= 1 - (X^2 + X^4 + X^3 + X + X^2 + X + X^2 + X^3) \\ &\quad + (X^6 + X^2 + X^4 + X^3 + X^4 + X^5) - (X^6) \\ &= 1 - 2X - 2X^2 - X^3 + X^4 + X^5 .\end{aligned} \tag{11.107}$$

There are 15 forward paths in the graph of Figure 11.15(b) (see Problem 11.8), and

$$\begin{aligned}\sum_i F_i \Delta_i &= X^5(1 - X - X^2 - X^3 + X^5) + X^4(1 - X^2) + X^6 \cdot 1 + X^5(1 - X^3) \\ &\quad + X^4 \cdot 1 + X^3(1 - X - X^2) + X^6(1 - X^2) + X^6 \cdot 1 + X^5(1 - X) \\ &\quad + X^8 \cdot 1 + X^4(1 - X - X^2 - X^3 + X^4) + X^7(1 - X^3) + X^6 \cdot 1 \\ &\quad + X^3(1 - X) + X^6 \cdot 1 \\ &= 2X^3 + X^4 + X^5 + X^6 - X^7 - X^8 .\end{aligned} \tag{11.108}$$

Hence, the codeword WEF is given by

$$\begin{aligned}A(X) &= \frac{2X^3 + X^4 + X^5 + X^6 - X^7 - X^8}{1 - 2X - 2X^2 - X^3 + X^4 + X^5} \\ &= 2X^3 + 5X^4 + 15X^5 + \cdots ,\end{aligned} \tag{11.109}$$

and this code contains two nonzero codewords of weight 3, five of weight 4, fifteen of weight 5, and so on.

Additional information about the weight properties of a code can be obtained using essentially the same procedure. If the modified state diagram is augmented by labeling each branch corresponding to a nonzero information block with W^w, where w is the weight of the k information bits on that branch, and labeling each branch with L, then the *codeword input–output weight enumerating function* (IOWEF) of the code is given by

$$A(W, X, L) = \sum_{w,d,l} A_{w,d,l} W^w X^d L^l . \tag{11.110}$$

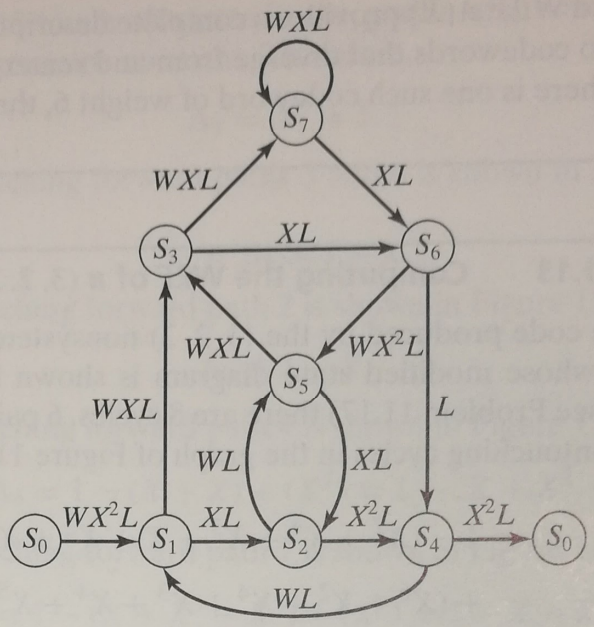


FIGURE 11.17: Augmented modified state diagram for the $(2, 1, 3)$ encoder of Figure 11.1.

The coefficient $A_{w,d,l}$ denotes the number of codewords with weight d, whose associated information weight is w, and whose length is l branches. The augmented modified state diagram for the encoder of Figure 11.1 is shown in Figure 11.17.

EXAMPLE 11.12 (Continued)

For the graph of Figure 11.17, it can be shown that (see Problem 11.18)

$$\begin{aligned}\Delta = 1 &- (X^8W^4L^7 + X^3W^3L^5 + X^7W^3L^6 + X^2W^2L^4 + X^4W^4L^7 + X^3W^3L^6 \\ &+ X^3WL^3 + XWL^2 + X^5W^3L^4 + X^4W^2L^3 + XWL) + (X^4W^4L^7 \\ &+ X^8W^4L^7 + X^3W^3L^6 + X^3W^3L^5 + X^4W^4L^7 + X^8W^4L^7 + X^7W^3L^6 \\ &+ X^4W^2L^4 + X^2W^2L^3 + X^5W^3L^4) - (X^4W^4L^7 + X^8W^4L^7) \\ = 1 &- XW(L + L^2) - X^2W^2(L^4 - L^3) - X^3WL^3 - X^4W^2(L^3 - L^4)\end{aligned} \tag{11.111}$$

and

$$\begin{aligned}\sum_i F_i\Delta_i = X^{12}W^4L^8 \cdot 1 &+ X^7W^3L^6(1 - XWL^2) + X^{11}W^3L^7(1 - XWL) \\ &+ X^6W^2L^5[1 - XW(L + L^2) + X^2W^2L^3] + X^8W^4L^8 \cdot 1 \\ &+ X^7W^3L^7(1 - XWL) + X^7WL^4(1 - XWL - X^4W^2L^3) \\ &= X^6W^2L^5 + X^7WL^4 - X^8W^2L^5.\end{aligned} \tag{11.112}$$

Hence, the codeword IOWEF is given by

$$\begin{aligned} A(W, X, L) &= \frac{X^6W^2L^5 + X^7WL^4 - X^8W^2L^5}{1 - XW(L + L^2) - X^2W^2(L^4 - L^3) - X^3WL^3 - X^4W^2(L^3 - L^4)} \\ &= X^6W^2L^5 + X^7(WL^4 + W^3L^6 + W^3L^7) \\ &\quad + X^8(W^2L^6 + W^4L^7 + W^4L^8 + 2W^4L^9) + \cdots . \end{aligned} \tag{11.113}$$

This implies that the codeword of weight 6 has length-5 branches and an information weight of 2, one codeword of weight 7 has length-4 branches and information weight 1, another has length-6 branches and information weight 3, the third has length-7 branches and information weight 3, and so on.

The term $WL^{\nu+1}$ in $A(W, X, L)$, corresponding to information weight 1, always represents the shortest path through the augmented modified state diagram. It is interesting to note, however, that this may not be the minimum-weight path. In Example 11.12, for instance, the term X^7WL^4 represents the shortest path, whereas the term $X^6W^2L^5$ represents the minimum-weight path. Also, it is important to note that the codeword WEF $A(X)$ is a property of the code; that is, it is invariant to the encoder representation. The codeword IOWEF $A(W, X, L)$, on the other hand, is a property of the encoder, since it depends on the mapping between input sequences and codewords (see Problem 11.21).

An alternative version of the IOWEF, which contains only information about the input and output weights but not the lengths of each codeword, can be obtained by simply setting the variable L in $A(W, X, L)$ to unity; that is,

$$A(W, X) = \sum_{w,d} A_{w,d} W^w X^d = A(W, X, L)|_{L=1}, \tag{11.114}$$

where $A_{w,d} = \sum_l A_{w,d,l}$ represents the number of codewords with weight d and information weight w. Similarly, the WEF $A(X)$ is related to the IOWEF as

$$A(X) = \sum_d A_d X^d = A(W, X)|_{W=1} = A(W, X, L)|_{W=L=1}, \tag{11.115}$$

where $A_d = \sum_w A_{w,d}$. In Chapter 16 we will see that an important tool needed to compute the WEF of turbo codes is the *codeword conditional weight enumerating function* (CWEF), which enumerates the weights of all codewords associated with particular information weights. For an information weight of w, the CWEF is given by

$$A_w(X) = \sum_d A_{w,d} X^d. \tag{11.116}$$

It follows directly from (11.114) that the IOWEF can be expressed as

$$A(W, X) = \sum_w W^w A_w(X). \tag{11.117}$$

Clearly, the CWEF is also a property of the encoder.

EXAMPLE 11.12 (Continued)

The simplified IOWEF of (11.113) becomes

$$A(W, X) = A(W, X, L)|_{L=1} = W^2X^6 + (W + 2W^3)X^7 + (W^2 + 4W^4)X^8 + \cdots, \tag{11.118}$$

indicating one codeword of weight 6 with information weight 2, three codewords of weight 7, one with information weight 1 and two with information weight 3, and so on, and we can obtain the WEF as

$$A(X) = A(W, X)|_{W=1} = X^6 + 3X^7 + 5X^8 + \cdots, \tag{11.119}$$

which agrees with the $A(X)$ computed previously in (11.106).

To compute the CWEFs for this encoder we must find the path weights associated with the information sequences of a given weight that traverse the modified state diagram from the initial state to the final state. There is one such information sequence of weight 1, $\mathbf{u} = (1\,0\,0\,0)$, three such sequences of weight 2, $\mathbf{u} = (1\,1\,0\,0\,0)$, $(1\,0\,1\,0\,0\,0)$, and $(1\,0\,0\,1\,0\,0\,0)$, nine such sequences of weight 3, $\mathbf{u} =$ $(1\,1\,1\,0\,0\,0)$, $(1\,1\,0\,1\,0\,0\,0)$, $(1\,0\,1\,1\,0\,0\,0)$, $(1\,0\,1\,0\,1\,0\,0\,0)$, $(1\,1\,0\,0\,1\,0\,0\,0)$, $(1\,0\,0\,1\,1\,0\,0\,0)$, $(1\,0\,1\,0\,0\,1\,0\,0\,0)$, $(1\,0\,0\,1\,0\,1\,0\,0\,0)$, and $(1\,0\,0\,1\,0\,0\,1\,0\,0\,0)$, and so on. (The last three 0's in each sequence represent the termination bits.) Thus, using Figure 11.17, we see that

$$A_1(X) = X^7, \tag{11.120a}$$

$$A_2(X) = X^6 + X^8 + X^{10}, \tag{11.120b}$$

and

$$A_3(X) = 2X^7 + 3X^9 + 3X^{11} + X^{13}. \tag{11.120c}$$

Finally, using (11.117), we can write

$$\begin{aligned} A(W, X) &= \sum_w W^w A_w(X) \\ &= WX^7 + W^2(X^6 + X^8 + X^{10}) + W^3(2X^7 + 3X^9 + 3X^{11} + X^{13}) + \cdots. \end{aligned} \tag{11.121}$$

Because of the different ways in which they were calculated, it is not immediately obvious that (11.118) and (11.121) give the same result; however, if we rewrite $A(W, X, L)$ in (11.113) listing the numerator and denominator terms in increasing powers of W, and then set $L = 1$, we obtain

$$\begin{aligned} A(W, X) &= \frac{WX^7 + W^2(X^6 - X^8)}{1 - W(2X + X^3)} \\ &= WX^7 + W^2(X^6 + X^8 + X^{10}) + W^3(2X^7 + 3X^9 + 3X^{11} + X^{13}) + \cdots, \end{aligned} \tag{11.122}$$

the same result as in (11.121).

The preceding example illustrates that if we wish to express the IOWEF in terms of increasing codeword weight, we should list the numerator and denominator terms in $A(W, X)$ in increasing powers of X prior to division; however, if we wish to express the IOWEF in terms of increasing information weight, we should list them in increasing powers of W.

Now, consider the case of terminated encoders, that is, encoders whose output sequences are limited to $\lambda = h + m$ blocks, where λ represents the total length of the input sequence, which comprises h information blocks and m termination blocks. To adapt the foregoing procedure for computing WEFs to terminated encoders, we write the numerator and denominator terms of the IOWEF $A(W, X, L)$ in increasing powers of L. Then, we perform division and drop terms of degree larger than L^{λ} from the resulting power series.

EXAMPLE 11.12 (Continued)

We begin by rewriting (11.113) as

$$A(W, X, L) = \frac{L^4WX^7 + L^5W^2(X^6 - X^8)}{1 - LWX - L^2WX - L^3[WX^3 + W^2(X^4 - X^2)] - L^4W^2(X^2 - X^4)}. \tag{11.123}$$

Now, let the encoder be terminated after $\lambda = 8$ blocks; that is, consider only input sequences with $h = 5$ information blocks and $m = 3$ termination blocks. Then, division and truncation of terms with degree larger than L^8 gives (see Problem 11.22)

$$\begin{aligned} A(W, X, L) = {} & L^4WX^7 + L^5W^2X^6 + L^6(W^2X^8 + W^3X^7) + L^7[W^2X^{10} \\ & + W^3(X^7 + X^{11}) + W^4X^8] + L^8[3W^3X^9 + W^4(X^8 + X^{10} + X^{12}) + W^5X^9] \end{aligned} \tag{11.124}$$

as the IOWEF of this terminated encoder. There are exactly 15 terms in this IOWEF, because we consider only information sequences whose first bit is a 1, and we do not consider the information sequence $\mathbf{u} = (1\ 0\ 0\ 0\ 1\ 0\ 0\ 0)$, since it diverges from and remerges with state S_0 twice. (Note that W^5 is the largest input weight represented in (11.124), since $\lambda - m = 5$.) We can now compute the simplified IOWEF, the CWEFs, and the WEF of this terminated encoder as

$$\begin{aligned} A(W, X) = A(W, X, L)|_{L=1} = {} & WX^7 + W^2(X^6 + X^8 + X^{10}) + W^3(2X^7 + 3X^9 + X^{11}) \\ & + W^4(2X^8 + X^{10} + X^{12}) + W^5X^9, \end{aligned} \tag{11.125a}$$

$$A_1(X) = X^7, \tag{11.125b}$$

$$A_2(X) = X^6 + X^8 + X^{10}, \tag{11.125c}$$

$$A_3(X) = 2X^7 + 3X^9 + X^{11}, \tag{11.125d}$$

$$A_4(X) = 2X^8 + X^{10} + X^{12}, \tag{11.125e}$$

$$A_5(X) = X^9, \tag{11.125f}$$

and

$$A(X) = A(W, X)|_{W=1} = X^6 + 3X^7 + 3X^8 + 4X^9 + 2X^{10} + X^{11} + X^{12}, \quad (11.125g)$$

respectively. Note that the CWEF $A_3(X)$ given in (11.125d) for the terminated encoder differs from the expression given in (11.120c) for the unterminated encoder. This is because some of the weight $w = 3$ information sequences in the unterminated case are more than $h = 5$ bits long, and thus they are not valid input sequences to the terminated encoder.

It is important to note here that the foregoing method for computing codeword WEFs considers only nonzero codewords that diverge from the all-zero state at the initial time unit and remerge exactly once. In other words, delayed versions of the same codeword or codewords that diverge and remerge more than once are not counted. (The reason for enumerating the weights of nonzero codewords in this fashion is related to the method used to calculate the bit-error probability of convolutional codes with maximum likelihood decoding, a subject that will be discussed in Chapter 12.) Thus, in the case of a terminated encoder with an information sequence of length $h = 5$ considered in the preceding example, only 15 nonzero codewords are counted, rather than the $2^5 - 1 = 31$ nonzero codewords that exist in the equivalent $(N, K^*) = [n(h + m),\ kh] = (16, 5)$ block code. Methods for calculating the complete WEF of the equivalent block code obtained by terminating a convolutional encoder are needed to compute the WEFs of turbo codes. These methods are discussed in more detail in Chapter 16.

To determine the information bit-error probability of convolutional codes with maximum likelihood decoding, a subject to be discussed in the next chapter, it is convenient to modify the codeword WEF expressions introduced previously to represent the number of nonzero information bits associated with codewords of a given weight. These modified expressions are referred to as *bit WEFs*. For an information weight of w, the *bit CWEF* is given by

$$B_w(X) = \sum_d B_{w,d} X^d, \quad (11.126)$$

where $B_{w,d} = (w/k)A_{w,d}$. Hence, $B_{w,d}$ can be interpreted as the total number of nonzero information bits associated with codewords of weight d produced by information sequences of weight w, divided by the number of information bits k per unit time. It follows directly that the *bit IOWEF* can be expressed as

$$B(W, X) = \sum_{w,d} B_{w,d} W^w X^d = \sum_w W^w B_w(X), \quad (11.127)$$

and the bit WEF is given by

$$B(X) = \sum_d B_d X^d = B(W, X)|_{W=1}, \quad (11.128)$$

where $B_d = \sum_w B_{w,d}$ is the total number of nonzero information bits associated with codewords of weight d, divided by the number of information bits k per unit

time. In this case, all three bit WEFs, $B_w(X)$, $B(W, X)$, and $B(X)$, are properties of the encoder.

We now note that the definitions of $A_{w,d}$ and $B_{w,d}$ given here imply that, for any w and d,

$$B_d = \sum_w B_{w,d} = \sum_w (w/k) A_{w,d}. \tag{11.129}$$

Equation (11.129) can then be used to establish the following formal relationship between the codeword IOWEF $A(W, X)$ and the bit WEF $B(X)$:

$$\begin{aligned} B(X) &= \sum_d B_d X^d = \sum_{w,d} (w/k) A_{w,d} X^d \\ &= \frac{1}{k} \frac{\partial [\sum_{w,d} A_{w,d} W^w X^d]}{\partial W}\bigg|_{W=1} = \frac{1}{k} \frac{\partial A(W, X)}{\partial W}\bigg|_{W=1}; \end{aligned} \tag{11.130}$$

that is, the bit WEF $B(X)$ can be calculated directly from the codeword IOWEF $A(W, X)$.

As will be seen in the next chapter, the decisions made by an MLD for convolutional codes at each time unit result in the decoding of k information bits. Thus, to determine the decoding error probability per information bit, we include a factor of $(1/k)$ in the bit WEFs defined here. For terminated convolutional codes, we will also consider maximum a posteriori probability decoding in the next chapter. This method of decoding is used, for example, in Chapter 16 on turbo coding. In this case, decoding decisions are made for the entire block of information bits at one time, and the WEFs are defined slightly differently. In particular, the IOWEF $A(W, X)$ is modified to include all $2^{K^*} - 1 = 2^{kh} - 1$ nonzero codewords that exist in the equivalent $(N, K^*) = [n(h + m), kh]$ block code, including delayed versions of a given codeword and codewords that diverge and remerge with the all-zero state more than once, and the corresponding codeword WEFs are modified accordingly. Also, if $K = k(h + m)$ represents the total number of input bits to the encoder, we define $B_{w,d} = (w/K)A_{w,d}$, and the corresponding bit WEFs are modified accordingly. Then, $B_{w,d}$ represents the total number of nonzero information bits associated with codewords of weight d produced by information sequences of weight w, divided by the total number of input bits K in a block. (Technically, since $K = k(h + m)$, only kh of the K input bits are information bits, but since h is normally large compared with m, we usually ignore this distinction.) These modified WEFs are discussed in more detail in Chapter 16.

EXAMPLE 11.12 (Continued)

Applying (11.130) to the codeword IOWEF given in (11.118) for the (2, 1, 3) encoder in this example, we obtain the bit WEF

$$\begin{aligned} B(X) &= \frac{1}{k} \frac{\partial A(W, X)}{\partial W}\bigg|_{W=1}, \\ &= \frac{\partial [W^2X^6 + (W + 2W^3)X^7 + (W^2 + 4W^4)X^8 + \cdots]}{\partial W}\bigg|_{W=1} \\ &= 2X^6 + 7X^7 + 18X^8 + \cdots. \end{aligned} \tag{11.131}$$

The coefficients of $B(X)$ in (11.131) represent the total number of nonzero information bits associated with codewords of each weight for this encoder.

We will see in the next chapter how the bit WEF $B(X)$ can be used to determine the bit-error probability associated with maximum likelihood decoding of convolutional codes.

For systematic encoders, since k information sequences and $(n-k)$ parity sequences comprise a codeword, the weight of a codeword is the information weight w plus the weight of the parity sequences. In this case, it is convenient to express the codeword CWEF as

$$A_w(Z) = \sum_z A_{w,z} Z^z, \tag{11.132}$$

where $A_{w,z}$ is the number of codewords with information weight w and *parity weight* z. (We note here that the number of codewords $A_{w,z}$ with information weight w and parity weight z is the same as the number of codewords $A_{w,d}$ with information weight w and total weight $d = z + w$.) Then, the *codeword input redundancy weight enumerating functions* (IRWEFs) are given by

$$A(W, Z, L) = \sum_{w,z,l} A_{w,z,l} W^w Z^z L^l, \tag{11.133}$$

where $A_{w,z,l}$ is the number of codewords of length l with information weight w and parity weight z, and

$$A(W, Z) = \sum_{w,z} A_{w,z} W^w Z^z = A(W, Z, L)|_{L=1} = \sum_{w,z} W^w A_w(Z). \tag{11.134}$$

It follows that the codeword WEF can be expressed as

$$A(X) = \sum_d A_d X^d = A(W, Z)|_{W=Z=X}, \tag{11.135}$$

where $A_d = \sum_{w+z=d} A_{w,z}$ is the total number of codewords of weight d. (The notation $\sum_{w+z=d} A_{w,z}$ implies that we are summing over all coefficients $A_{w,z}$ such that $w + z = d$.) The bit WEFs for systematic encoders are determined in the same way as in (11.126)–(11.128), with $B_{w,z} = (w/k)A_{w,z}$ (or, for terminated encoders, with $B_{w,z} = (w/K)A_{w,z}$). For an information weight of w, the *bit CWEF* is given by

$$B_w(Z) = \sum_z B_{w,z} Z^z, \tag{11.136}$$

and $B_{w,z}$ can be interpreted as the total number of nonzero information bits associated with codewords of parity weight z and information weight w, divided by the number of information bits k per unit time. It follows directly that the *bit IRWEF* can be expressed as

$$B(W, Z) = \sum_{w,z} B_{w,z} W^w Z^z = \sum_w W^w B_w(Z), \tag{11.137}$$

and the *bit WEF* is given by

$$B(X) = \sum_d B_d X^d = B(W, Z)|_{W=Z=X}, \tag{11.138}$$

where $B_d = \sum_{w+z=d} B_{w,z}$ is the total number of nonzero information bits associated with codewords of weight d, divided by the number of information bits k per unit time. Finally, (11.137) and (11.138) can be used to develop the following formal relationship between the codeword IRWEF $A(W, Z)$ and the bit WEF $B(X)$:

$$\begin{aligned} B(X) = B(W, Z)|_{W=Z=X} &= \sum_{w,z} B_{w,z} W^w Z^z|_{W=Z=X} = \sum_{w,z} (w/k) A_{w,z} W^w Z^z|_{W=Z=X} \\ &= \frac{1}{k} W \frac{\partial[\sum_{w,z} A_{w,z} W^w Z^z]}{\partial W}\bigg|_{W=Z=X} = \frac{1}{k} W \frac{\partial A(W, Z)}{\partial W}\bigg|_{W=Z=X}; \end{aligned} \tag{11.139}$$

that is, the bit WEF $B(X)$ can be calculated directly from the codeword IRWEF $A(W, Z)$.

Again, for systematic encoders, the codeword WEF is a property of the code and the codeword CWEFs and IRWEFs, and the bit CWEFs, IRWEFs, and WEF are all properties of the encoder.

To determine the IRWEF of a systematic encoder using the transfer function method introduced here, we replace the labels X^d representing codeword weights in the augmented modified state diagram with labels Z^z representing parity weights. We now illustrate the techniques for finding WEFs for systematic encoders with an example using a systematic feedback encoder. First, we note that for feedback encoders, the bits needed to terminate the encoder are, in general, nonzero. Thus, in this case, we refer to the powers of W in the WEFs as nonzero input bits rather than nonzero information bits, since termination bits are not considered information bits; however, this distinction is important only for low-weight input sequences.

EXAMPLE 11.14 Computing the WEFs of a (2, 1, 2) Systematic Feedback Encoder

Consider the (2, 1, 2) systematic feedback convolutional encoder whose generator matrix is given by

$$\mathbf{G}(D) = [1 \quad (1 + D^2)/(1 + D + D^2)]. \tag{11.140}$$

The controller canonical form realization and augmented modified state diagram for this encoder are shown in Figure 11.18. Following the same procedure as in Examples 11.12 and 11.13, we see that there are three cycles in the graph of Figure 11.18(b):

$$\begin{aligned} &\text{Cycle 1: } S_1 S_3 S_2 S_1 \quad (C_1 = Z^2 L^3) \\ &\text{Cycle 2: } S_1 S_2 S_1 \quad (C_2 = W L^2) \\ &\text{Cycle 3: } S_3 S_3 \quad (C_3 = W L) \end{aligned}$$

We also see that there is one pair of nontouching cycles:

$$\text{Cycle Pair 1: (Cycle 2, Cycle 3)} \quad (C_2 C_3 = W^2 L^3).$$

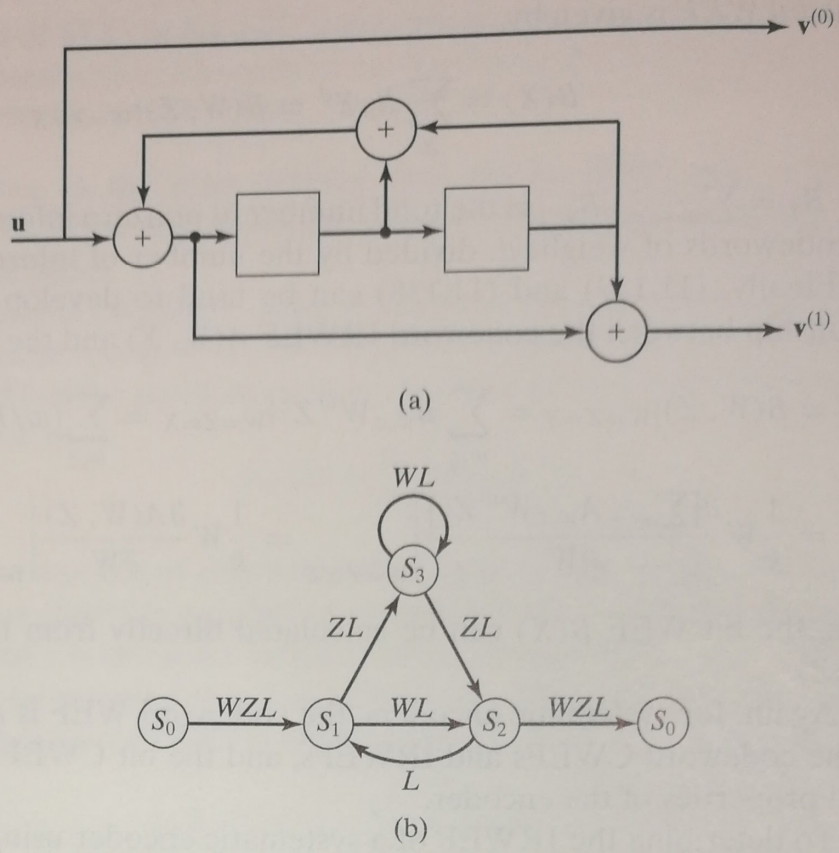


FIGURE 11.18: (a) The controller canonical form realization and (b) the augmented modified state diagram for a (2, 1, 2) systematic feedback encoder.

In this case there are no other sets of nontouching cycles. Therefore,

$$\Delta = 1 - (Z^2L^3 + WL^2 + WL) + (W^2L^3) = 1 - Z^2L^3 - W(L + L^2) + W^2L^3. \quad (11.141)$$

We now see that there are two forward paths in the graph of Figure 11.18(b):

$$\text{Forward Path 1: } S_0S_1S_2S_0 \qquad (F_1 = W^3Z^2L^3)$$
$$\text{Forward Path 2: } S_0S_1S_3S_2S_0 \qquad (F_2 = W^2Z^4L^4)$$

The subgraph not touching forward path 1 is simply the single branch cycle around state S_3, that is, cycle 3. Therefore

$$\Delta_1 = 1 - WL. \quad (11.142)$$

Forward path 2 touches all states in the graph, and hence,

$$\Delta_2 = 1. \quad (11.143)$$

Finally, applying (11.99), we obtain the codeword IRWEF:

$$\begin{aligned} A(W, Z, L) &= \frac{W^3Z^2L^3(1 - WL) + W^2Z^4L^4 \cdot 1}{1 - Z^2L^3 - W(L + L^2) + W^2L^3} \\ &= \frac{W^2Z^4L^4 + W^3Z^2L^3 - W^4Z^2L^4}{1 - Z^2L^3 - W(L + L^2) + W^2L^3} \quad (11.144) \\ &= \frac{W^2Z^4L^4}{F(Z, L)} + W^3\left[\frac{Z^2L^3}{F(Z, L)} + \frac{Z^4(L^5 + L^6)}{F^2(Z, L)}\right] + \cdots, \end{aligned}$$

where we have defined $F(Z, L) = 1 - Z^2L^3$. In this case we see that there are an infinite number of codewords corresponding to input sequences of weight 2. (In the state diagram, cycle 1 has zero input weight associated with it, so that an infinite number of codewords with input weight 2 can be generated by traversing this cycle an arbitrary number of times.) We can represent these codewords by expanding the first term in (11.144) as follows:

$$\frac{W^2Z^4L^4}{F(Z, L)} = W^2(Z^4L^4 + Z^6L^7 + Z^8L^{10} + \cdots), \tag{11.145}$$

that is, corresponding to weight-2 input sequences, there are codewords with parity weight $4 + 2j$ and length $4 + 3j$, $j = 0, 1, 2, \cdots$, owing to the gain $C_1 = Z^2L^3$ of cycle 1. Similarly, there are an infinite number of codewords corresponding to input sequences of weight 3, and so on.

We can now obtain the IRWEF without length information from (11.144) as follows:

$$\begin{aligned} A(W, Z) = A(W, Z, L)|_{L=1} &= \frac{W^2Z^4 + W^3Z^2 - W^4Z^2}{1 - Z^2 - 2W + W^2} \\ &= \frac{W^2Z^4}{F(Z)} + W^3\left[\frac{Z^2}{F(Z)} + \frac{2Z^4}{F^2(Z)}\right] + \cdots, \end{aligned} \tag{11.146}$$

where $F(Z) = 1 - Z^2$, and we can use (11.134) to obtain the following CWEFs:

$$A_2(Z) = \frac{Z^4}{1 - Z^2}, \tag{11.147a}$$

$$A_3(Z) = \frac{Z^2}{1 - Z^2} + \frac{2Z^4}{1 - 2Z^2 + Z^4}, \tag{11.147b}$$

and so on. It is interesting to note here that (11.147a) implies that all input weight-2 codewords have a total weight of at least 6 (input weight 2 plus parity weight of at least 4), whereas (11.147b) implies that there is one input weight-3 codeword with a total weight of only 5. Thus, the minimum-weight codeword in this case is produced by a weight-3 input sequence. Finally, the codeword WEF is obtained by setting $W = Z = X$ in (11.146), giving the expression

$$\begin{aligned} A(X) = A(W, Z)|_{W=Z=X} &= \frac{X^6 + X^5 - X^6}{1 - X^2 - 2X + X^2} \\ &= \frac{X^5}{1 - 2X} = X^5 + 2X^6 + 4X^7 + 8X^8 + \cdots. \end{aligned} \tag{11.148}$$

(Note that the substitution $W = Z = X$ is made directly into the rational function representation of (11.146) rather than into the series representation in order of increasing input weight, since this results in a simplified expression for $A(X)$.) We see from (11.148) that this code contains one codeword of weight 5, two of weight 6, four of weight 7, and so on. In Problem 11.23 it is shown that the equivalent nonsystematic feedforward encoder, which produces the same code as

the systematic feedback encoder of (11.140), has the same codeword WEF $A(X)$ as the one in (11.148). But its IOWEFs and CWEFs represent different input–output relationships than the IRWEFs and CWEFs computed previously, because these functions are properties of the encoder.

Finally, following (11.136)–(11.138), the bit CWEFs are given by

$$B_2(Z) = \frac{2Z^4}{1 - Z^2}, \tag{11.149a}$$

$$B_3(Z) = \frac{3Z^2}{1 - Z^2} + \frac{6Z^4}{1 - 2Z^2 + Z^4}, \tag{11.149b}$$

the bit IRWEF is given by

$$\begin{aligned} B(W, Z) &= \sum_w W^w B_w(Z) \\ &= \frac{2W^2Z^4}{1 - Z^2} + \frac{3W^3Z^2}{1 - Z^2} + \frac{6W^3Z^4}{1 - 2Z^2 + Z^4} + \cdots, \end{aligned} \tag{11.150}$$

and the bit WEF is given by

$$\begin{aligned} B(X) &= B(W, Z)|_{W=Z=X} \\ &= \frac{3X^5}{1 - X^2} + \frac{2X^6}{1 - X^2} + \frac{6X^7}{1 - 2X^2 + X^4} + \cdots. \end{aligned} \tag{11.151}$$

We can also use (11.139) to calculate $B(X)$ as follows:

$$\begin{aligned} B(X) &= \frac{1}{k} W \frac{\partial A(W, Z)}{\partial W}\bigg|_{W=Z=X} \\ &= W \frac{\partial[(W^2Z^4 + W^3Z^2 - W^4Z^2)/(1 - Z^2 - 2W + W^2)]}{\partial W}\bigg|_{W=Z=X} \\ &= \frac{N(W, Z)}{D(W, Z)}\bigg|_{W=Z=X}, \end{aligned} \tag{11.152}$$

where

$$\begin{aligned} N(W, Z) = {} & 2W^2(Z^4 - Z^6) + W^3(3Z^2 - 5Z^4) - 2W^4(Z^2 - 2Z^4 + 3Z^2) \\ & + 7W^5Z^2 - 2W^6Z^2 \end{aligned} \tag{11.153a}$$

and

$$D(W, Z) = 1 - 2Z^2 + Z^4 - 4W(1 - Z^2) + 2W^2(3 - Z^2) - 4W^3 + W^4. \tag{11.153b}$$

Now, carrying out the division we obtain (see Problem 11.24)

$$B(X) = 3X^5 + 6X^6 + 14X^7 + 32X^8 + \cdots; \tag{11.154}$$

that is, the weight-5 codeword has information weight 3, the two weight-6 codewords have total information weight 6, and so on. (Because $k = 1$ in this case,

the coefficients of $B(X)$ represent the total information weight associated with all codewords of a given weight.) We note that (11.151) and (11.154) do not appear to give the same result, because the three terms shown in (11.151) are based entirely on input weights 2 and 3. If input weights 4, 5, and 6 are also included, then the coefficients of $B(X)$ in (11.151) and (11.154) would agree up to powers of X^8. In general, the method based on (11.139) gives the simplest way to calculate $B(X)$.

In Example 11.14 we saw that there was a zero-input weight cycle in the state diagram that allowed the encoder to generate an infinite number of codewords with input weight 2. These zero-input weight cycles are characteristic of all feedback encoders, in which a single input 1 drives the encoder away from the all-zero state, and then, after the zero-input weight cycle is traversed some number of times, another input 1 returns the encoder to the all-zero state. We will see in Chapter 16 that the codewords produced by these weight-2 input sequences play a critical role in the performance of turbo codes.

We now introduce an alternative method of computing the WEFs of convolutional encoders. The foregoing method, based on representing the state diagram of the encoder as a signal flow graph, has the advantage of an easy visualization of the set of all codewords that diverge from and remerge with the all-zero state exactly once. Computationally, though, it becomes difficult to apply once the overall constraint length ν exceeds 3, that is, once the state diagram contains more than 8 states. A more efficient method of computing the transfer function of an encoder is to use a state variable representation of the encoding equations. We illustrate the technique by continuing the preceding example.

EXAMPLE 11.14 (Continued)

In the augmented modified state diagram of Figure 11.18, let Σ_i be a state variable that represents the weights of all paths from the initial state S_0 to the state S_i, $i = 1, 2, 3$. Then, we can write the following set of state equations:

$$\Sigma_1 = WZL + L\Sigma_2, \tag{11.155a}$$

$$\Sigma_2 = WL\Sigma_1 + ZL\Sigma_3, \tag{11.155b}$$

$$\Sigma_3 = ZL\Sigma_1 + WL\Sigma_3, \tag{11.155c}$$

and the overall transfer function of the graph is given by

$$A(W, Z, L) = WZL\Sigma_2. \tag{11.156}$$

Equations (11.155) are a set of linear equations in the state variables Σ_1, Σ_2, and Σ_3. We can rewrite them in matrix form as

$$\begin{bmatrix} WZL \\ 0 \\ 0 \end{bmatrix} = \begin{bmatrix} 1 & -L & 0 \\ -WL & 1 & -ZL \\ -ZL & 0 & 1-WL \end{bmatrix} \begin{bmatrix} \Sigma_1 \\ \Sigma_2 \\ \Sigma_3 \end{bmatrix}. \tag{11.157}$$

Using Cramer's rule to solve (11.157) for the state variable Σ_2, we obtain

$$\Sigma_2 = \frac{\det\begin{bmatrix} 1 & WZL & 0 \\ -WL & 0 & -ZL \\ -ZL & 0 & 1-WL \end{bmatrix}}{\det\begin{bmatrix} 1 & -L & 0 \\ -WL & 1 & -ZL \\ -ZL & 0 & 1-WL \end{bmatrix}} = \frac{WZ^3L^3 + W^2ZL^2 - W^3ZL^3}{1 - Z^2L^3 - W(L+L^2) + W^2L^3}. \tag{11.158}$$

Now, we apply (11.156) to find the IRWEF:

$$A(W, Z, L) = WZL\Sigma_2 = \frac{W^2Z^4L^4 + W^3Z^2L^3 - W^4Z^2L^4}{1 - Z^2L^3 - W(L+L^2) + W^2L^3}, \tag{11.159}$$

which is the same result obtained using the signal flow graph approach.

It should be clear from the preceding examples that the transfer function approach to finding the WEFs of a convolutional encoder quickly becomes impractical as the overall constraint length ν becomes large. As will be seen in the next section, however, the important distance properties of a code necessary to estimate its performance at moderate-to-large SNRs can be computed without the aid of the transfer function. Nevertheless, the transfer function remains an important conceptual tool, and it will be used in Chapters 12, 16, and 18 to obtain performance bounds for convolutional codes used with maximum likelihood decoding.

11.3 DISTANCE PROPERTIES OF CONVOLUTIONAL CODES

The performance of a convolutional code depends on the decoding algorithm employed and the distance properties of the code. In this section several distance measures for convolutional codes are introduced. Their relation to code performance will be discussed in later chapters.

The most important distance measure for convolutional codes is the minimum *free distance* d_{free}.

Definition 11.7 The minimum free distance of a convolutional code is defined as

$$d_{free} \triangleq \min_{\mathbf{u}', \mathbf{u}''} \{d(\mathbf{v}', \mathbf{v}'') : \mathbf{u}' \neq \mathbf{u}''\}, \tag{11.160}$$

where $\mathbf{v}'$ and $\mathbf{v}''$ are the codewords corresponding to the information sequences $\mathbf{u}'$ and $\mathbf{u}''$, respectively.

In (11.160) it is assumed that the codewords $\mathbf{v}'$ and $\mathbf{v}''$ have finite length and start and end in the all-zero state S_0; that is, m termination blocks are appended to the information sequences $\mathbf{u}'$ and $\mathbf{u}''$. If $\mathbf{u}'$ and $\mathbf{u}''$ have different lengths, zeros are added

to the shorter sequence so that the corresponding codewords have equal lengths. Hence, d_{free} is the minimum distance between any two finite-length codewords in the code. Because a convolutional code is a linear code,

$$\begin{aligned} d_{free} &= \min_{\mathbf{u}', \mathbf{u}''} \{w(\mathbf{v}' + \mathbf{v}'') : \mathbf{u}' \neq \mathbf{u}''\} \\ &= \min_{\mathbf{u}} \{w(\mathbf{v}) : \mathbf{u} \neq \mathbf{0}\} \\ &= \min_{\mathbf{u}} \{w(\mathbf{uG}) : \mathbf{u} \neq \mathbf{0}\}, \end{aligned} \tag{11.161}$$

where $\mathbf{v}$ is the codeword corresponding to the information sequence $\mathbf{u}$. Hence, d_{free} is the minimum-weight codeword produced by any finite-length nonzero information sequence. Also, it is the minimum weight of all finite-length paths in the state diagram that diverge from and remerge with the all-zero state S_0, and hence it is the lowest power of X in the WEF $A(X)$. For example, $d_{free} = 6$ for the $(2, 1, 3)$ code of Example 11.12, and $d_{free} = 3$ for the $(3, 2, 2)$ code of Example 11.13.

Another important distance measure for convolutional codes is the *column distance function* (CDF). Let

$$[\mathbf{v}]_l = (v_0^{(0)} v_0^{(1)} \cdots v_0^{(n-1)}, v_1^{(0)} v_1^{(1)} \cdots v_1^{(n-1)}, \cdots, v_l^{(0)} v_l^{(1)} \cdots v_l^{(n-1)}) \tag{11.162}$$

denote the lth truncation of the codeword $\mathbf{v}$, and let

$$[\mathbf{u}]_l = (u_0^{(1)} u_0^{(2)} \cdots u_0^{(k)}, u_1^{(1)} u_1^{(2)} \cdots u_1^{(k)}, \cdots, u_l^{(1)} u_l^{(2)} \cdots u_l^{(k)}) \tag{11.163}$$

denote the lth truncation of the information sequence $\mathbf{u}$.

Definition 11.5 The column distance function of order l, d_l, is defined as

$$\begin{aligned} d_l &\triangleq \min_{[\mathbf{u}']_l, [\mathbf{u}'']_l} \{d([\mathbf{v}']_l, [\mathbf{v}'']_l) : [\mathbf{u}']_0 \neq [\mathbf{u}'']_0\} \\ &= \min_{[\mathbf{u}]_l} \{w[\mathbf{v}]_l) : [\mathbf{u}]_0 \neq \mathbf{0}\}, \end{aligned} \tag{11.164}$$

where $\mathbf{v}$ is the codeword corresponding to the information sequence $\mathbf{u}$.

Hence, d_l is the minimum-weight codeword over the first $(l + 1)$ time units whose initial information block is nonzero. In terms of the generator matrix of the code,

$$[\mathbf{v}]_l = [\mathbf{u}]_l [\mathbf{G}]_l, \tag{11.165}$$

where $[\mathbf{G}]_l$ is a $k(l+1) \times n(l+1)$ matrix consisting of the first $n(l+1)$ columns of $\mathbf{G}$ (see 11.22) with the following form:

$$[\mathbf{G}]_l = \begin{bmatrix} \mathbf{G}_0 & \mathbf{G}_1 & \cdots & \mathbf{G}_l \\ & \mathbf{G}_0 & \cdots & \mathbf{G}_{l-1} \\ & & \ddots & \vdots \\ & & & \mathbf{G}_0 \end{bmatrix}, \quad l \leq m, \tag{11.166a}$$

$$[\mathbf{G}]_l = \begin{bmatrix} \mathbf{G}_0 & \mathbf{G}_1 & \cdots & \cdots & \mathbf{G}_{m-1} & \mathbf{G}_m & & & & \\ & \mathbf{G}_0 & \cdots & \cdots & \mathbf{G}_{m-2} & \mathbf{G}_{m-1} & \mathbf{G}_m & & & \\ & & \ddots & \vdots & \vdots & \vdots & \vdots & \ddots & & \\ & & & \ddots & \mathbf{G}_1 & \mathbf{G}_2 & \mathbf{G}_3 & \cdots & \mathbf{G}_m & \\ & & & & \mathbf{G}_0 & \mathbf{G}_1 & \mathbf{G}_2 & \cdots & \mathbf{G}_{m-1} & \mathbf{G}_m \\ & & & & & \mathbf{G}_0 & \mathbf{G}_1 & \cdots & \mathbf{G}_{m-2} & \mathbf{G}_{m-1} \\ & & & & & & \mathbf{G}_0 & \vdots & \vdots & \vdots \\ & & & & & & & \ddots & \mathbf{G}_1 & \mathbf{G}_2 \\ & & & & & & & & \mathbf{G}_0 & \mathbf{G}_1 \\ & & & & & & & & & \mathbf{G}_0 \end{bmatrix}, \quad l > m. \tag{11.166b}$$

Then,

$$d_l = \min_{[\mathbf{u}]_l} \{ w([\mathbf{u}]_l [\mathbf{G}]_l) : [\mathbf{u}]_0 \neq \mathbf{0} \} \tag{11.167}$$

is seen to depend only on the first $n(l+1)$ columns of $\mathbf{G}$. This accounts for the name "column distance function."

We note here that the form of the generator matrix $\mathbf{G}$ used in (11.166) (and in (11.22)) assumes a feedforward encoder realization, that is, a polynomial generator matrix with memory order m. For feedback encoders with rational function generator matrices, the associated generator sequences have infinite duration, and the $k \times n$ submatrices $\mathbf{G}_l$ extend infinitely to the right in (11.166). In this case, the CDF can still be expressed using (11.167). We also note that the free distance d_{free} is independent of the encoder realization, that is, it is a code property, whereas the column distance function d_l is an encoder property; however, if we restrict our attention to encoders for which the $k \times n$ submatrix $\mathbf{G}_0$ has full rank, then the CDF d_l is independent of the encoder realization. (This results in the sensible property that a codeword is not delayed with respect to its information sequence.) Thus, we will refer to the CDF d_l as a code property.

Definition 11.8 implies that d_l cannot decrease with increasing l; that is, it is a monotonically nondecreasing function of l. The most efficient way of computing the CDF of a code is to modify one of the sequential decoding algorithms to be introduced in Chapter 13 (see Section 13.4). The complete CDF of the $(2, 1, 16)$ nonsystematic code with $\mathbf{G}(D) = [1 + D + D^2 + D^5 + D^6 + D^8 + D^{13} + D^{16},\ 1 + D^3 + D^4 + D^7 + D^9 + D^{10} + D^{11} + D^{12} + D^{14} + D^{15} + D^{16}]$ is shown in Figure 11.19.

Two cases are of specific interest: $l = m$ and $l \to \infty$. For $l = m$, d_m is called the *minimum distance* of a convolutional code. Hence, d_m is also denoted by d_{min}. From (11.167) we see that d_{min} represents the minimum-weight codeword over the first $(m+1)$ time units whose initial information block is nonzero. For the code of Figure 11.19, $d_{min} = d_{16} = 9$. Much of the early work in convolutional codes treated d_{min} as the distance parameter of most interest, because the principal decoding techniques at that time had a decoding memory of $(m+1)$ time units. (See Section 13.5 on majority-logic decoding of convolutional codes.) More recently, as maximum

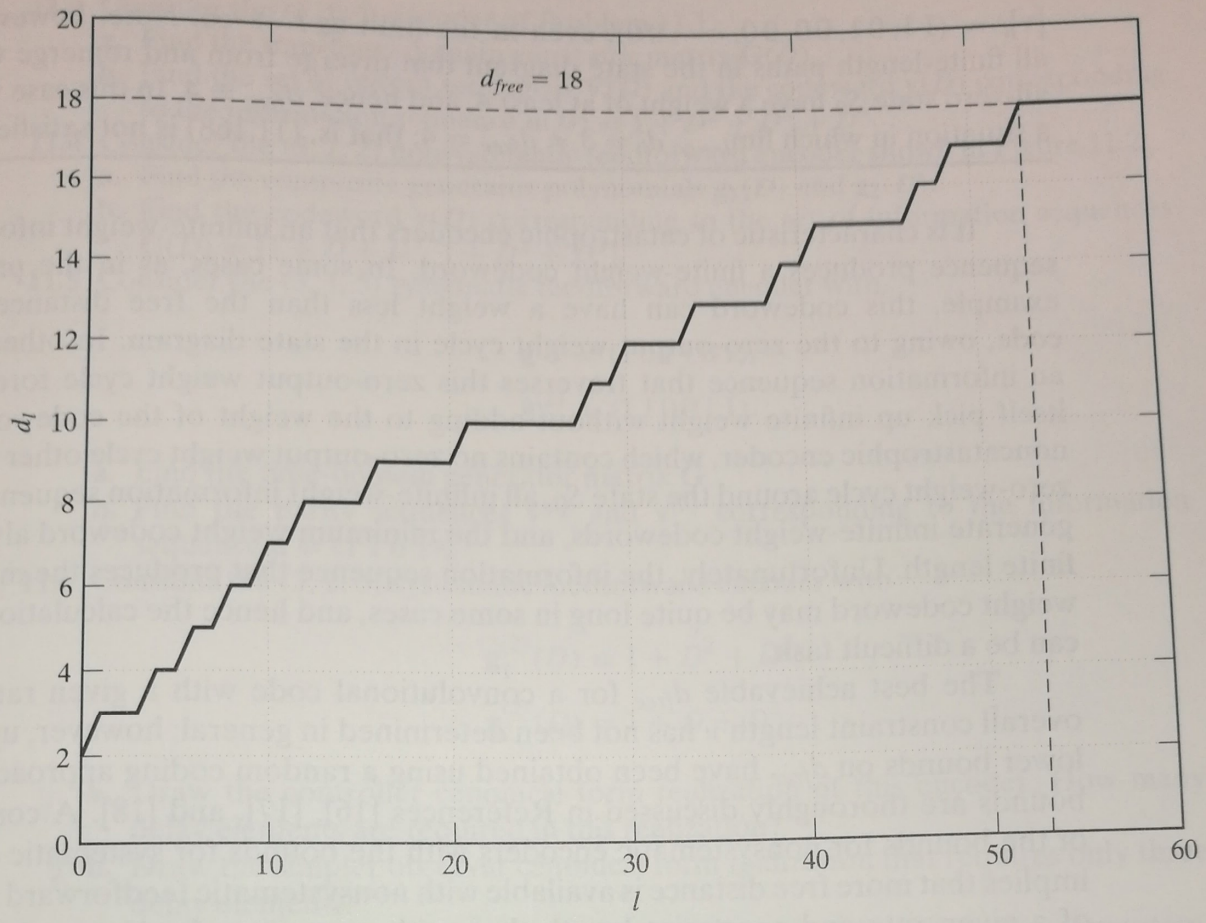


FIGURE 11.19: The column distance function of a (2, 1, 16) code.

likelihood (ML) decoding, maximum a posteriori (MAP) decoding, and sequential decoding have become more prominent, d_{free} and the CDF have replaced d_{min} as the distance parameters of primary interest, since the decoding memory of these techniques is unlimited. A thorough discussion of the relationship between distance measures and decoding algorithms is included in Chapters 12 and 13.

For $l \to \infty$, $\lim_{l\to\infty} d_l$ is the minimum-weight codeword of any length whose first information block is nonzero. Comparing the definitions of $\lim_{l\to\infty} d_l$ and d_{free}, we can show that (see Problem 11.31) for noncatastrophic encoders

$$\lim_{l\to\infty} d_l = d_{free}. \tag{11.168}$$

Hence, d_l eventually reaches d_{free}, and then it stays constant. This usually happens when l reaches about 3ν. For the code of Figure 11.19 with $\nu = 16$, $d_l = 18$ for $l \geq 53$, and hence, $d_{free} = 18$. Because d_{free} is defined for finite-length codewords, however, (11.168) is not necessarily true in the case of catastrophic encoders.

EXAMPLE 11.10 (Continued)

Consider again the catastrophic encoder whose state diagram is shown in Figure 11.14. For this encoder, $d_0 = 2$, and $d_1 = d_2 = \cdots = \lim_{l\to\infty} d_l = 3$, since the truncated information sequence $[\mathbf{u}]_l = (1, 1, 1, \cdots, 1)$ always produces the truncated codeword

$[\mathbf{v}]_l = (11, 01, 00, 00, \cdots, 00)$, even in the limit as $l \to \infty$. Note, however, that all finite-length paths in the state diagram that diverge from and remerge with the all-zero state S_0 have a weight of at least 4, and hence, $d_{free} = 4$. In this case we have a situation in which $\lim_{l\to\infty} d_l = 3 \neq d_{free} = 4$; that is, (11.168) is not satisfied.

It is characteristic of catastrophic encoders that an infinite-weight information sequence produces a finite-weight codeword. In some cases, as in the preceding example, this codeword can have a weight less than the free distance of the code, owing to the zero-output weight cycle in the state diagram. In other words, an information sequence that traverses this zero-output weight cycle forever will itself pick up infinite weight without adding to the weight of the codeword. In a noncatastrophic encoder, which contains no zero-output weight cycle other than the zero-weight cycle around the state S_0, all infinite-weight information sequences must generate infinite-weight codewords, and the minimum weight codeword always has finite length. Unfortunately, the information sequence that produces the minimum-weight codeword may be quite long in some cases, and hence the calculation of d_{free} can be a difficult task.

The best achievable d_{free} for a convolutional code with a given rate R and overall constraint length ν has not been determined in general; however, upper and lower bounds on d_{free} have been obtained using a random coding approach. These bounds are thoroughly discussed in References [16], [17], and [18]. A comparison of the bounds for nonsystematic encoders with the bounds for systematic encoders implies that more free distance is available with nonsystematic feedforward encoders of a given rate and constraint length than with systematic feedforward encoders. This observation is verified by the code construction results presented in the next two chapters and has important consequences when a code with large d_{free} must be selected for use with ML, MAP, or sequential decoding. Thus, if a systematic encoder realization is desired, it is usually better to select a nonsystematic feedforward encoder with large d_{free} and then convert it to an equivalent systematic feedback encoder.

PROBLEMS

11.1 Consider the (3, 1, 2) nonsystematic feedforward encoder with

$$\mathbf{g}^{(0)} = (1\,1\,0),$$
$$\mathbf{g}^{(1)} = (1\,0\,1),$$
$$\mathbf{g}^{(2)} = (1\,1\,1).$$

a. Draw the encoder block diagram.
b. Find the time-domain generator matrix $\mathbf{G}$.
c. Find the codeword $\mathbf{v}$ corresponding to the information sequence $\mathbf{u} = (1\,1\,1\,0\,1)$.

11.2 Consider the (4, 3, 3) nonsystematic feedforward encoder shown in Figure 11.3.
a. Find the generator sequences of this encoder.
b. Find the time-domain generator matrix $\mathbf{G}$.
c. Find the codeword $\mathbf{v}$ corresponding to the information sequence $\mathbf{u} = (110, 011, 101)$.

11.3 Consider the (3, 1, 2) encoder of Problem 11.1.

a. Find the transform-domain generator matrix $\mathbf{G}(D)$.

b. Find the set of output sequences $\mathbf{V}(D)$ and the codeword $\mathbf{v}(D)$ corresponding to the information sequence $\mathbf{u}(D) = 1 + D^2 + D^3 + D^4$.

11.4 Consider the (3, 2, 2) nonsystematic feedforward encoder shown in Figure 11.2.

a. Find the composite generator polynomials $\mathbf{g}_1(D)$ and $\mathbf{g}_2(D)$.

b. Find the codeword $\mathbf{v}(D)$ corresponding to the set of information sequences $\mathbf{U}(D) = [1 + D + D^3, 1 + D^2 + D^3]$.

11.5 Consider the (3, 1, 5) systematic feedforward encoder with

$$\mathbf{g}^{(1)} = (1\,0\,1\,1\,0\,1),$$

$$\mathbf{g}^{(2)} = (1\,1\,0\,0\,1\,1).$$

a. Find the time-domain generator matrix $\mathbf{G}$.

b. Find the parity sequences $\mathbf{v}^{(1)}$ and $\mathbf{v}^{(2)}$ corresponding to the information sequence $\mathbf{u} = (1\,1\,0\,1)$.

11.6 Consider the (3, 2, 3) systematic feedforward encoder with

$$\mathbf{g}_1^{(2)}(D) = 1 + D^2 + D^3,$$

$$\mathbf{g}_2^{(2)}(D) = 1 + D + D^3.$$

a. Draw the controller canonical form realization of this encoder. How many delay elements are required in this realization?

b. Draw the simpler observer canonical form realization that requires only three delay elements.

11.7 Verify the sequence of elementary row operations leading from the nonsystematic feedforward realizations of (11.34) and (11.70) to the systematic feedback realizations of (11.66) and (11.71).

11.8 Draw the observer canonical form realization of the generator matrix $\mathbf{G}'(D)$ in (11.64) and determine its overall constraint length ν.

11.9 Consider the rate $R = 2/3$ nonsystematic feedforward encoder with generator matrix

$$\mathbf{G}(D) = \begin{bmatrix} D & D & 1 \\ 1 & D^2 & 1 + D + D^2 \end{bmatrix}.$$

a. Draw the controller canonical form encoder realization for $\mathbf{G}(D)$. What is the overall constraint length ν?

b. Find the generator matrix $\mathbf{G}'(D)$ of the equivalent systematic feedback encoder. Is $\mathbf{G}'(D)$ realizable? If not, find an equivalent realizable generator matrix and draw the corresponding minimal encoder realization. Is this minimal realization in controller canonical form or observer canonical form? What is the minimal overall constraint length ν?

11.10 Use elementary row operations to convert the rate $R = 2/3$ generator matrix of (11.77) to systematic feedback form, and draw the minimal observer canonical form encoder realization. Find and draw a nonsystematic feedback controller canonical form encoder realization with the same number of states.

11.11 Redraw the observer canonical form realization of the (3, 2, 2) systematic feedback encoder in Figure 11.7(b) using the notation of (11.82) and the relabeling scheme of Figure 11.11.

11.12 Consider the (3, 1, 2) systematic feedback encoder shown in Figure 11.6(c). Determine the $\nu = 2$ termination bits required to return this encoder to the all-zero state when the information sequence $\mathbf{u} = (10111)$.

11.13 Consider the (4, 3, 3) nonsystematic feedforward encoder realization in controller canonical form shown in Figure 11.3.

a. Draw the equivalent nonsystematic feedforward encoder realization in observer canonical form, and determine the number of termination bits required to return this encoder to the all-zero state. What is the overall constraint length of this encoder realization?

b. Now, determine the equivalent systematic feedback encoder realization in observer canonical form, and find the number of termination bits required to return this encoder to the all-zero state. What is the overall constraint length of this encoder realization?

11.14 Consider the (2, 1, 2) nonsystematic feedforward encoder with $\mathbf{G}(D) = [1 + D^2 \;\; 1 + D + D^2]$.

a. Find the GCD of its generator polynomials.

b. Find the transfer function matrix $\mathbf{G}^{-1}(D)$ of its minimum-delay feedforward inverse.

11.15 Consider the (2, 1, 3) nonsystematic feedforward encoder with $\mathbf{G}(D) = [1 + D^2 \;\; 1 + D + D^2 + D^3]$.

a. Find the GCD of its generator polynomials.

b. Draw the encoder state diagram.

c. Find a zero-output weight cycle in the state diagram.

d. Find an infinite-weight information sequence that generates a codeword of finite weight.

e. Is this encoder catastrophic or noncatastrophic?

11.16 Find the general form of transfer function matrix $\mathbf{G}^{-1}(D)$ for the feedforward inverse of an (n, k, ν) systematic encoder. What is the minimum delay l?

11.17 Verify the calculation of the WEF in Example 11.13.

11.18 Verify the calculation of the IOWEF in Example 11.12.

11.19 Consider the (3, 1, 2) encoder of Problem 11.1.

a. Draw the state diagram of the encoder.

b. Draw the modified state diagram of the encoder.

c. Find the WEF $A(X)$.

d. Draw the augmented modified state diagram of the encoder.

e. Find the IOWEF $A(W, X, L)$.

11.20 Using an appropriate software package, find the WEF $A(X)$ for the (4, 3, 3) encoder of Figure 11.3.

11.21 Consider the equivalent systematic feedback encoder for Example 11.1 obtained by dividing each generator polynomial by $\mathbf{g}^{(0)}(D) = 1 + D^2 + D^3$.

a. Draw the augmented modified state diagram for this encoder.

b. Find the IRWEF $A(W, Z)$, the two lowest input weight CWEFs, and the WEF $A(X)$ for this encoder.

c. Compare the results obtained in (b) with the IOWEF, CWEFs, and WEF computed for the equivalent nonsystematic feedforward encoder in Example 11.1.

11.22 Verify the calculation of the IOWEF given in (11.124) for the case of a terminated convolutional encoder.

11.23 Consider the equivalent nonsystematic feedforward encoder for Example 11.14 obtained by multiplying $\mathbf{G}(D)$ in (11.140) by $\mathbf{g}^{(0)}(D) = 1 + D + D^2$.

a. Draw the augmented modified state diagram for this encoder.
b. Find the IOWEF $A(W, X, L)$, the three lowest input weight CWEFs, and the WEF $A(X)$ for this encoder.
c. Compare the results obtained in (b) with the IRWEF, CWEFs, and WEF computed for the equivalent systematic feedback encoder in Example 11.14.

11.24 In Example 11.14, verify all steps leading to the calculation of the bit WEF in (11.154).

11.25 Consider the $(2, 1, 2)$ systematic feedforward encoder with $\mathbf{G}(D) = [1 \;\; 1 + D^2]$.
a. Draw the augmented modified state diagram for this encoder.
b. Find the IRWEF $A(W, Z, L)$, the three lowest input weight CWEFs, and the WEF $A(X)$ for this encoder.

11.26 Recalculate the IOWEF $A(W, X, L)$ in Example 11.12 using the state variable approach of Example 11.14.

11.27 Recalculate the WEF $A(X)$ in Example 11.13 using the state variable approach of Example 11.14.

11.28 Consider the $(3, 1, 2)$ code generated by the encoder of Problem 11.1.
a. Find the free distance d_{free}.
b. Plot the complete CDF.
c. Find the minimum distance d_{min}.

11.29 Repeat Problem 11.28 for the code generated by the encoder of Problem 11.15.

11.30 **a.** Prove that the free distance d_{free} is independent of the encoder realization, i.e., it is a code property.
b. Prove that the CDF d_l is independent of the encoder realization; that is, it is a code property. (Assume that the $k \times n$ submatrix $\mathbf{G}_0$ has full rank.)

11.31 Prove that for noncatastrophic encoders

$$\lim_{l \to \infty} d_l = d_{free}.$$

BIBLIOGRAPHY

1. P. Elias, "Coding for Noisy Channels," *IRE Conv. Rec., p. 4*: 37–47, 1955.

2. J. M. Wozencraft and B. Reiffen, *Sequential Decoding*, MIT Press, Cambridge, 1961.

3. J. L. Massey, *Threshold Decoding*. MIT Press, Cambridge, 1963.

4. A. J. Viterbi, "Error Bounds for Convolutional Codes and an Asymptotically Optimum Decoding Algorithm," *IEEE Trans. Inform. Theory*, IT-13: 260–69, April 1967.

5. L. R. Bahl, J. Cocke, F. Jelinek, and J. Raviv, "Optimal Decoding of Linear Codes for Minimal Symbol Error Rate," *IEEE Trans. Inform. Theory*, IT-20: 284–87, March 1974.

6. G. Ungerboeck and I. Csajka, "On Improving Data-Link Performance by Increasing the Channel Alphabet and Introducing Sequence Coding," in *IEEE International Symposium on Information Theory (ISIT 1976) Book of Abstracts*, p. 53, Ronneby, Sweden, June 1976.

7. G. Ungerboeck, "Channel Coding with Multilevel/Phase Signals," *IEEE Trans. Inform. Theory*, IT-28: 55–67, January 1982.

8. C. Berrou, A. Glavieux, and P. Thitimajshima, "Near Shannon Limit Error-Correcting Coding and Decoding: Turbo Codes," in *Proc. IEEE International Conference on Communications (ICC 93)*, 1064–70, Geneva, Switzerland, May 1993.

9. C. Berrou and A. Glavieux, "Near Optimum Error Correcting and Decoding: Turbo-codes," *IEEE Trans. Commun.*, COM-44: 1261–71, October 1996.

10. A. Dholakia, *Introduction to Convolutional Codes*. Kluwer Academic, Dordrecht, The Netherlands, 1994.

11. L. H. C. Lee, *Convolutional Coding: Fundamentals and Applications*. Norwood, Mass., Artech House, 1997.

12. R. Johannesson and K. S. Zigangirov, *Fundamentals of Convolutional Coding*. IEEE Press, Piscataway, N.J., 1999.

13. G. D. Forney Jr., "Convolutional Codes I: Algebraic Structure," *IEEE Trans. Inform. Theory*, IT-16: 720–38, Nov. 1970.

14. J. L. Massey and M. K. Sain, "Inverses of Linear Sequential Circuits," *IEEE Trans. Computers*, C-17: 330–37, April 1968.

15. S. Mason and H. Zimmermann, *Electronic Circuits, Signals, and Systems*. John Wiley, New York, N.Y., 1960.

16. R. Johannesson and K. S. Zigangirov, "Distances and Distance Bounds for Convolutional Codes," in *Topics in Coding Theory—In Honour of Lars H. Zetterberg, G. Einarsson et al., eds.*, 109–36, Springer-Verlag, Berlin, 1989.

17. G. D. Forney Jr., "Convolutional Codes II: Maximum Likelihood Decoding," *Inform. and Control*, 25: 222–66, July 1974.

18. D. J. Costello Jr., "Free Distance Bounds for Convolutional Codes," in *IEEE Trans. Inform. Theory*, IT-20: 356–65, May 1974.