# 19

# Spectrum Control

*Coding*, which refers to the translation between the user-provided information bits (source bits) and the transmitted data symbols (coded symbols), Error-control coding, whose aim is to mitigate the effects of noise, was discussed in Chapters 12 and 13. This chapter discusses the use of coding to control the statistics of the data symbols, thereby introducing a measure of control over the spectrum of the transmitted signal. For example, undesired correlations among information bits can be removed by *scrambling*, which is a reversible transformation of the bits in a way that affects the statistics. Alternatively, the spectrum can be controlled by introducing a controlled correlation among data symbols in the form of *redundancy* (the remaining sections). In Chapters 12 and 13 we saw applications of redundancy to the correction and prevention of channel errors.

One way to control the spectrum is through the design of a *line code* (Sections 19.2 and 19.3). One major motivation in baseband systems is the problem of *baseline wander* introduced by the a.c. coupling inherent in transformers and broadband amplifiers. This phenomenon is described in Section 19.1. In Section 19.2 a number of different types of line codes for baseband systems are described, most of them oriented toward control of baseline wander. Then in Section 19.3, we describe a different class of techniques based on introducing spectral nulls at arbitrary frequencies using filtering. The resulting ISI is mitigated using transmitter precoding (Chapter 8). An important special case is called *partial response*. A related class of techniques that achieve high bandwidth efficiency and immunity to nonlinearities in bandpass systems, called *continuous-phase modulation*, is described in Section 19.4.

Often we want to ensure that the transmitted signal has sufficient *randomness* or activity so that timing recovery and other functions can be reliably performed. For example, because people type relatively slowly, a computer terminal transmits null characters most of the time. A long sequence of null characters results in a highly correlated line signal. Such signals can foil timing recovery (Chapter 16), adaptive equalization (Chapter 9) and echo cancellation

(Chapter 20), all of which assume that the transmitted symbols are uncorrelated. *Scrambling*, described in Section 19.5, is intended to *remove* strong correlations among the information bits so as to make them appear more random.

## 19.1. GOALS OF LINE CODES

In earlier chapters we saw examples of different signaling schemes, such as binary antipodal and orthogonal signaling. These are simple examples of line codes. Line codes can be used to accomplish desirable goals such as to place spectral nulls at particular frequencies. A common goal is to introduce a null in the spectrum at d.c., thereby enabling transmission of a baseband PAM waveform over a channel that cannot accommodate a d.c. component in the data signal. An alternative for such a channel would be to use passband PAM, but in many applications baseband PAM in conjunction with line coding is a more cost-effective alternative.

Line coding is not as big an issue in passband systems as baseband, for several reasons. Baseband systems, particularly those operating over cable, typically have a large variation in attenuation over the Nyquist bandwidth and also a large variation in crosstalk coupling loss (Section 18.2). Hence, there is much that can be done to improve the performance of these systems by control of the transmitted power spectrum. Passband systems, in contrast, usually have a relatively constant attenuation vs. frequency (since the bandwidth is narrow relative to the center frequency), and crosstalk coupling may not be an issue or is relatively frequency-independent. In this chapter we will therefore limit our discussion of line codes to baseband PAM systems.

A common consideration in the line code is the tradeoff between symbol rate and the number of transmitted levels. As discussed in Section 5, the symbol rate relates directly to the required channel bandwidth, while the number of levels relates directly to the noise immunity. The line code also affects the transmitted power spectrum and hence the crosstalk into foreign systems (as in wire-pair or radio transmission) and radio-frequency interference (RFI). The line code affects many aspects of the implementation, such as the complexity of the equalization, detection, echo cancellation, and timing recovery circuitry.

A side benefit of some line codes is the ability to perform *in-service monitoring* of the data signal to be sure that the system is performing well even while transmitting the information bits. The coder adds redundancy to the transmitted data symbols, and the receiver checks to see that the code constraints are preserved after detection. Line coding can also be used to make the digital transmission signal more immune to nonlinearities, such as those on satellite and radio channels (Section 19.4).

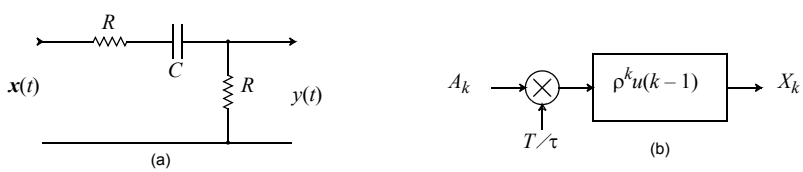



**Fig. 19-1.** a. An a.c. coupled circuit. b. A discrete-time system characterizing the postcursor ISI introduced by baseline wander in this circuit.

Important properties of line codes include their redundancy, their running digital sum, and their power spectrum.

### 19.1.1. Redundancy

If the number of distinct transmitted symbols is $L$, and the symbol rate is $f_b$ symbols per second, then the information-carrying capacity is

$$R = f_b \cdot \log_2 L \quad \text{bits/sec} . \tag{19.1}$$

Define $B$ as the information bit rate provided to the user. When $B = R$, there is no redundancy in the code, and our only degree of freedom in the design of the code is the choice of the deterministic translation between information bits and transmitted data symbols, described in detail in Section 5.

**Example 19-1.** If $L = 4$, without redundancy we can assign two information bits to each data symbol. There are 4! = 24 possible ways in which we can assign these two bits to the four distinct data symbols.

For many practical line codes, $B < R$. The difference between $B$ and $R$ represents a *redundancy* that can be put to good use. If the information bits are statistically independent, and $B = R$, then the transmitted data symbols must also be independent. But by allowing redundancy, we can make the transmitted data symbols statistically dependent, regardless of the statistics of the information, and hence exercise some control over the power spectrum of the transmitted signal.

### 19.1.2. Running Digital Sum

Many baseband systems use transformer coupling or a.c.-coupled electronics, which implies that the channel has infinite loss at d.c. In a sense the channel is actually passband, although it is special in that the highpass cutoff frequency is small relative to the symbol rate. Line coding techniques can deal with this situation, in place of more complicated carrier modulation techniques. Line coding also allows us to concentrate the signal power at frequencies near d.c., where the cable attenuation is often the lowest. Actually, as we will see, some line coding techniques are actually closely related to carrier modulation.

The effect of a.c. coupling on a channel is a form of intersymbol interference (ISI) called *baseline wander*. This effect, for the a.c. coupling circuit in Fig. 19-1a (or of a transformer, which is similar), on an input PAM signal $\sum_k A_k \delta(t - kT)$ is analyzed in Problem 19-1. The conclusion is that the equivalent discrete-time channel is approximately characterized by the equivalent system generating the postcursor ISI in Fig. 19-1b, where $\tau = 2RC$ is the time constant and $\rho = e^{-T/\tau}$. This undesired *baseline wander* ISI, a consequence of the zero at d.c. in the channel response, is a major consideration in the choice of a line code.

There are two things needed to minimize the baseline wander problem. First, we must make the time constant $\tau$ large (or a.c. coupling cutoff frequency small), since this will minimize the $T/\tau$ term. When $\tau$ is large, $\rho \approx 1$, and the ISI at sample $k$ is approximately

$$X_k \approx \frac{T}{\tau} S_{k-1} , \qquad S_k = \sum_{m=-\infty}^{k} A_m , \tag{19.2}$$

where $S_k$ is called the *running digital sum (RDS)*. The RDS is an important property of the line code [1] because it predicts accurately the magnitude of the baseline wander ISI for a low cutoff frequency. Further, it is easily shown [2] that when the RDS is bounded, there is a spectral null at d.c.; conversely, for spectra generated by finite-state machines with a null at d.c., the RDS is bounded [3]. The second design consideration for the line code is therefore to ensure that the RDS $S_k$ is small. For a line code not explicitly designed to minimize baseline wander, the RDS could grow to infinity. In this case the ISI in Fig. 19-1b would not actually grow to infinity (Problem 19-2) since $\rho < 1$, but it could become quite large.

### 19.1.3. Transmitted Power Spectrum

The transmitted power spectrum is given by (3.84), which we repeat here for convenience. The PAM signal given by

$$X(t) = \sum_{k=-\infty}^{\infty} A_k g(t - kT) \tag{19.3}$$

is not wide-sense stationary, even if it is assumed that the transmitted data symbols $A_k$ are wide-sense stationary. For some purposes, however, it is permissible to randomize the phase epoch in (19.3); this yields wide-sense stationarity, and thus the power spectrum is (Appendix 3-A)

$$S_X(f) = \frac{1}{T} |G(f)|^2 S_A(e^{j2\pi fT}) \ . \tag{19.4}$$

The line code (along with the statistics of the source data) determines $S_A(e^{j2\pi fT})$, and the pulse shape chosen influences the power spectrum through the $|G(f)|^2$ term.

## 19.2. LINE CODE OPTIONS

Dozens of line codes have been seriously proposed. This section gives a representative set of these codes [4][5].

### 19.2.1. Linear Line Codes

*Linear line codes* are those in which the transmitted data symbols depend linearly on the information bits. In this subsection we will discuss three such line codes: the *binary antipodal* code, the *twinned binary* code, and *alternate mark inversion (AMI)*.

#### Binary Antipodal Codes

The simplest line codes transmit a pulse or its negative to send a “zero” or “one” bit, respectively (binary antipodal signaling). Several commonly used pulse shapes for binary antipodal signaling are shown in Fig. 19-2.

The only way to ensure no d.c. content for all possible transmitted data symbol sequences is to choose a pulse shape with no d.c. content (that is, $G(0) = 0$). The only pulse shape in Fig. 19-2 with this property is the *biphase* or *Manchester* pulse [6]. For the other two pulse shapes, RZ and NRZ, we must somehow insure that the average rates of positive and negative pulses are equal (we will see ways of doing this shortly). The use of biphase is simpler, but in spite of the zero in the spectrum at d.c. for this pulse, there will still be baseline wander due to

the low-frequency attenuation of the a.c. coupling. In other words, while the integral of the pulse is zero, the convolution of the pulse with a decaying exponential will have small but non-zero area. The magnitude of this source of intersymbol interference is easily predicted for the biphase (or any other) pulse shape.

**Exercise 19-1.** Define $\beta$ as the ratio of the cutoff frequency of the a.c. coupling to the symbol rate. Show that for the biphase code, the intersymbol interference from the last transmitted symbol has magnitude

$$(1 - e^{-\pi\beta})^2 . \tag{19.5}$$

Thus, as the cutoff frequency decreases, this intersymbol interference goes to zero.

**Example 19-2.** If we require that the intersymbol interference from the last symbol be down by a factor of $10^{-2}$, then $\beta = 0.033$; i.e., the cutoff frequency must be 3.3% of the symbol rate.

The biphase line code and a similar code known as the *Wal2 pulse shape* ( Problem 19-3) are often chosen for their "self-equalizing" properties, meaning that a single compromise equalizer will suffice for a wide range of line lengths in wire-pair and coax systems. The intuitive reason for this is that since the response to a single positive-going pulse has a long tail, and $g(t)$ consists of a positive-going pulse followed immediately by a negative-going pulse, the tail of the negative pulse will tend to cancel the tail from the positive pulse.

Choosing a pulse shape with zero integral is an effective and simple solution to the baseline wander problem. In addition, the fact that every symbol interval has a zero crossing in the center simplifies timing recovery (Chapter 16). However, we pay a high price for this zero crossing, since the high-frequency energy in the transmitted signal is larger. It is shown in Problem 19-4 that any pulse with no d.c. content that obeys the Nyquist criterion must have at least 100% excess bandwidth, or twice the minimum bandwidth of pulses that are allowed to have non-zero integral.

The biphase pulse shape can be viewed in two alternative ways which will suggest other ways to eliminate the d.c. component of a data waveform.

- Start with a binary antipodal code with NRZ pulses, and multiply the resulting signal by a square wave with period equal to the symbol interval. We can think of this square wave as a carrier modulation, approximately centering the signal at the symbol rate. Since the bandwidth of the signal before modulation can easily be less than the symbol rate, this modulation avoids a d.c. content in the signal. Thus, biphase can be viewed as a simple passband PAM scheme. This interpretation also explains why the resulting biphase data signal bandwidth following equalization is roughly twice as great as for NRZ or RZ.
- Increase the symbol rate to twice the incoming bit rate, and use binary antipodal signaling with an NRZ pulse shape (at this higher symbol rate). Follow each information

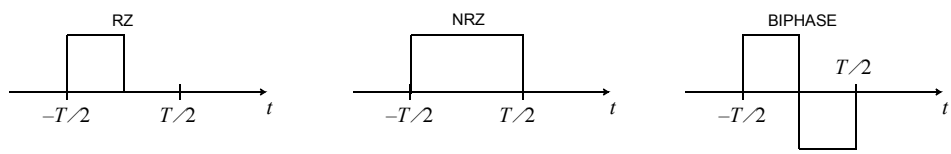



**Fig. 19-2.** Return-to-zero (RZ), non-return-to-zero (NRZ), and biphase pulses.

bit by another bit which is its complement; the result is a biphase pulse shape. For example, information bits (...0011...) would become (...01011010...), at double the symbol rate. Thus, we can view biphase as being equivalent to NRZ, with redundancy added by doubling the symbol rate.

The biphase or similar pulse shape is a good choice where implementation simplicity is desirable and the distance between transmitter and receiver is modest, as in a local-area network (Chapter 1). But where we want to increase range by limiting the signal bandwidth, we will find better alternatives. These alternatives minimize baseline wander by more sophisticated means, and allow us to concentrate the signal power at lower frequencies where cable attenuation and crosstalk are lesser problems.

**Twinned Binary Code**

We can improve on biphase at the expense of additional implementation complexity by coding the transmitted data symbols. In other words, we can introduce a zero at d.c. in $S_A(^{j2\pi fT})$ rather than $G(f)$ by forcing the transmitted data symbols to be correlated. The only way that redundancy can be introduced without increasing the symbol rate is by increasing the number of levels. In a *pseudoternary line code*, we use a three-level data symbol to transmit one bit of information. The redundancy inherent in transmitting only one bit of information with three levels can be used to accomplish many goals, including the reduction of baseline wander. Whereas with the biphase code we paid a price of greater signal bandwidth, with pseudoternary line codes we suffer a reduction in noise immunity; i.e., for the same peak power level a smaller noise level will cause an error.

An example of such a code is the *twinned binary code* invented by Meacham [7]. This code is not practical, but is easy to understand and can be made practical by a simple modification shown later. If the transmitted information bits are designated $b_k$ and assume the values "0" and "1", then the transmitted symbols are

$$a_k = b_k - b_{k-1} \,. \tag{19.6}$$

It is easy to verify that $a_k$ assumes the three values $\{-1, 0, +1\}$. For simplicity, throughout the remainder of this section we will denote the levels of a ternary code by "+","0", and "–", with the obvious association to the actual pulse amplitudes. Because only one bit of information is conveyed per data symbol, the code is pseudoternary. In fact, we can make the following association: for a positive transition in the bit stream (from "0" to "1") transmit a "+", for a negative transition transmit a "–", and for no transition transmit a "0". From the power spectrum,

$$S_A(e^{j2\pi fT}) = S_B(e^{j2\pi fT})|1 - e^{-j2\pi fT}|^2 = 4S_B(e^{j2\pi fT})\sin^2(\pi fT) \,, \tag{19.7}$$

we see that a zero has been introduced into the spectrum at d.c. (and at all multiples of the symbol rate, since the spectrum is periodic in the symbol rate).

The twinned binary code is illustrated in Fig. 19-3. We will denote the delay operator by $D$ rather than $z^{-1}$ in this and the following two chapters, as is conventional in the coding literature. In the receiver we have a ternary slicer; that is, a slicer appropriate for a ternary signal, with decision thresholds at $\pm 1/2$. The decoder, which follows the slicer, simply implements the transfer function $(1 - D)^{-1}$, which is the reciprocal of the encoder transfer function $(1 - D)$. The idea here is similar to the DFE in Chapter 8 — we introduce intersymbol interference in the coder, and eliminate it using past decisions in the decoder. There is a major difference however,

in that the decoder follows the slicer (hard decoding) rather than surrounding the slicer as in the DFE, and thus we do not gain the noise immunity advantages of the DFE. This approach has the same problem as the DFE; namely, error propagation results if the encoder and decoder ever get into different states due to decision errors.

**Example 19-3.** Observe what happens if we have a long sequence of 1's followed by 0's on the input data stream. Then the input and output of the coder are shown below:

$$\begin{array}{cccccccccccc} 0 & 1 & 1 & 1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 \\ & 1 & 0 & 0 & 0 & 0 & 0 & -1 & 0 & 0 & 0 & 0 & 0 \end{array} \quad . \tag{19.8}$$

Notice that the only distinguishing characteristic between a sequence of 0's and 1's is the polarity of the coder output at the *beginning* of the sequence. In the receiver, a single decision error at that point can easily cause a sequence of 1's to be turned into a sequence of 0's, or *vice versa*.

Actually the error propagation displayed in (19.8) is fundamental to the $(1 - D)$ response of the channel. Because this channel response passes no d.c., strings of inputs of the same polarity tend to be confused at the channel output. We saw this same phenomenon for the ML sequence detector on the same channel in Example 9-38. In that case, the sequence detector had a infinite number of minimum-distance error events corresponding to sequences of inputs of the same polarity. We will see a simple solution to this problem in the next subsection.

The redundancy inherent in transmitting one bit of information per ternary symbol can be used for in-service monitoring. This is a major advantage shared by all pseudoternary codes. All combinations of ternary digits are not possible; any impossible combination at the slicer output indicates that an error has been made. For example, two "+"'s in a row, even with an arbitrary number of intervening 0's, is impossible because this would indicate two positive transitions in a row in the input bit stream $b_k$. In fact, we can state the redundancy constraint rather concisely. Any number of 0's in a row are allowed; every non-zero symbol must have the opposite polarity from the last non-zero symbol. This latter property ensures that there is no d.c. content in the transmitted signal. It also follows that the RDS is bounded by either $0 \le S_k \le 1$ or $-1 \le S_k \le 0$, depending on the polarity of the first non-zero pulse transmitted. This implies that the *digital sum variation (DSV)*, the difference between the largest and smallest RDS, is unity for this line code. This is the smallest possible DSV for a pseudoternary code.

It is important to note the distinction between the biphase code and the twinned binary code. The former introduces a zero at d.c. by changing the transmitted pulse shape, and in the process, boosting the high frequencies in the signal. The transmitted signal still has two possible levels at any time. In the twinned binary code, we are able to introduce the zero

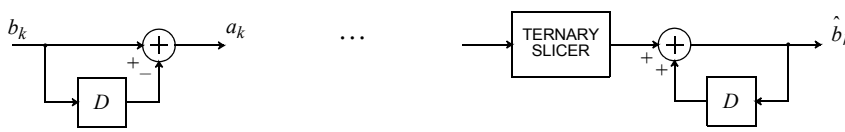



**Fig. 19-3.** Coding and decoding for the twinned binary code. D is the delay operator, equivalent to $z^{-1}$.

without changing the transmitted pulse, but the price we pay is an increase in the number of levels to three, a need for a ternary or three-level slicer, and reduced noise immunity for the same peak transmitted power.

We can improve the spectral properties of the twinned binary code by *two-way interleaving*. To do this, replace (19.6) by

$$a_k = b_k - b_{k-2} \,. \tag{19.9}$$

**Exercise 19-2.** Show that the power spectrum of (19.9) has a null at both d.c. and at half the symbol rate, $f = 1/(2T)$.

The advantage of having a null in the spectrum at half the symbol rate is that it can make practical a system with zero excess bandwidth without the requirement for "brickwall" lowpass filters. We will elaborate on this point in Section 19.3. But for our present purpose, it is valuable to understand another interpretation of (19.9), namely as two interleaved and independent twinned binary encoders. This interpretation follows from rewriting (19.9) as separate equations, one for even and one for odd-numbered data symbols,

$$a_{2k} = b_{2k} - b_{2(k-1)} \,, \qquad a_{2k+1} = b_{2k+1} - b_{2(k-1)+1} \,. \tag{19.10}$$

Observe that these two equations are independent, since one involves only even-numbered input bits and the other odd-numbered bits. This leads to the interpretation of Fig. 19-4. The input bit stream is decimated to two half-rate streams, and each is applied to its own line coder. For our present example, each line coder happens to be a twinned binary coder.

The interleaved configuration of Fig. 19-4 remarkably always leads to a spectral null at half the symbol rate if the constituent line coders have spectral nulls at d.c. This is because whenever the transmitted data symbols have a spectral null at d.c., they must also have a spectral null at the symbol rate (because of the periodicity of the sampled-data power spectrum). Since in Fig. 19-4 each coder is operating at half the symbol rate, the spectrum of each must have a null at half the symbol rate. The superposition of their outputs preserves that property! This implies that we can get a half-symbol rate spectral null starting with any favorite line code simply by interleaving, although we will always pay the price of doubling of the RDS, with a resultant increase in baseline wander.

**Exercise 19-3.** Argue that the RDS of the interleaved line coders of Fig. 19-4 will be double the RDS of the constituent line code. Hence, for the twinned binary code, the RDS will fall in the range $-2 \le RDS \le 2$. What is the DSV?

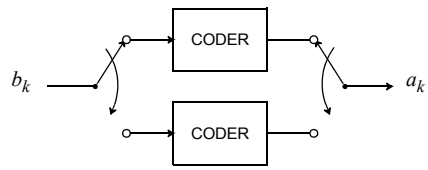



**Fig. 19-4.** Two interleaved line coders.

### AMI Line Code

The error propagation problem of the twinned binary code can be overcome using *precoding*, similar to the transmitter precoding of Section 8.1; the result is known as *alternate mark inversion (AMI)* or a *bipolar* line code. This line code was invented by Barker and is commercially and historically important because it was used in the first commercial PCM system, the T1-Carrier system designed by Bell Laboratories in 1962 [8].

Error propagation occurs in the twinned binary code because the $(1 - D)$ response causes an ambiguity for long strings of input 0's or 1's. In particular, these two cases are easily confused with one another. We can remove this ambiguity by a form of *differential encoding* of the bit stream prior to application to the line coder. A precoder and postcoder are shown in Fig. 19-5, where the symbol "⊕" has the special meaning of *modulo-two summation*. For binary inputs, the modulo-two summation is the same as an *exclusive-or* circuit, with truth table given below:

| Inputs | Output |
|---|---|
| 0 0 | 0 |
| 0 1 | 1 |
| 1 0 | 1 |
| 1 1 | 0 |

The precoder function can be represented by the equation

$$c_k = b_k \oplus c_{k-1}\,, \tag{19.11}$$

where $c_k$ is also a bit (assuming values "0" and "1").

The precoder and postcoder recover the original bit, since from Fig. 19-5 the output is

$$b_k \oplus c_{k-1} \oplus c_{k-1} = b_k\,. \tag{19.12}$$

This strange looking result follows from

$$c_{k-1} \oplus c_{k-1} = 0\,, \tag{19.13}$$

as the reader can readily verify.

The complete AMI coder is shown in Fig. 19-6; it is simply a combination of the twinned binary coder and the differential precoder. There are two different kinds of adders in this diagram, a normal adder and a modulo-two adder. For clarity, two distinct delays have been used where in fact one would suffice.

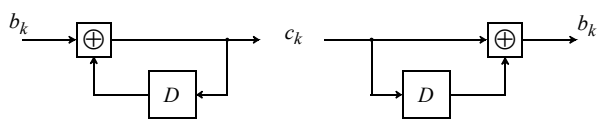



**Fig. 19-5.** A differential precoder and postcoder for the AMI line code.

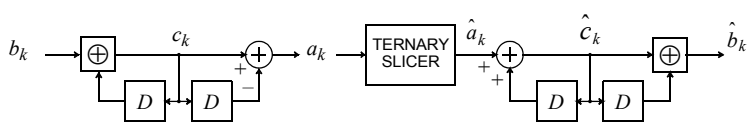



**Fig. 19-6.** An AMI coder realized by precoder followed by $(1 - D)$ filter.

The precoder is called differential because it can be described as follows: a "zero" bit at the input is transmitted as no change in the precoder output bit, and a "one" bit is transmitted as a change in the output bit (either from one to zero or zero to one). This removes the ambiguity that caused the error propagation because sequences of 0's and 1's at the precoder output both correspond to 0's at the precoder input.

**Example 19-4.** Repeating Example 19-3 with the precoder inserted, we get the following sequences at the input of the precoder, output of the precoder, and output of the $(1 - D)$ filter:

| | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 1 | 1 | 1 | 1 | 1 | 1 | 0 | 0 | 0 | 0 | 0 |
| 0 | 1 | 0 | 1 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 |
| | + | – | + | – | + | – | 0 | 0 | 0 | 0 | 0 |

This assumes that the precoder output was initially zero. Strings of 1's are transformed into alternating plus-minus, which easily passes through the a.c. coupled channel.

This precoding idea will be generalized in Section 19.3, where we will see that AMI is a special case of *partial response* coding.

Another interpretation of the absence of error propagation is that the decoder function is *memoryless*, i.e. the current input bit can be determined from the ternary slicer output without regard to the past (equivalently, the decoder has no internal state). To verify the memoryless property, write the decoder mathematically as

$$\hat{c}_k = \hat{a}_k + \hat{c}_{k-1} \tag{19.14}$$

$$b_k = \hat{c}_k \oplus \hat{c}_{k-1} = (\hat{a}_k + \hat{c}_{k-1}) \oplus \hat{c}_{k-1} \quad . \tag{19.15}$$

It will turn out that in (19.15), $b_k$ is a function only of $\hat{a}_k$ and not of $\hat{c}_{k-1}$. Consider for example the case $\hat{a}_k = 1$. Then for $\hat{c}_{k-1} = 0$,

$$(1 + \hat{c}_{k-1}) \oplus \hat{c}_{k-1} = (1 + 0) \oplus 0 = 1 \oplus 0 = 1 \tag{19.16}$$

and similarly for $\hat{c}_{k-1} = 1$,

$$(1 + \hat{c}_{k-1}) \oplus \hat{c}_{k-1} = (1 + 1) \oplus 1 = 2 \oplus 1 = 1 \quad . \tag{19.17}$$

Considering the other two cases, we can build the following truth table for the decoder:

| $\hat{a}_k$ | $\hat{b}_k$ |
|---|---|
| + | 0 |
| 0 | 1 |
| – | 1 |

To reiterate, the decoder output is a memoryless function of the slicer output, so the decoder has no internal state to get out of synchronization with the encoder, and there can be no error propagation. Based on this simplified descriptions of the encoder and decoder, we can state the AMI line code succinctly: encode a "zero" as a "0" transmitted symbol, and code a "one" alternately as "+" and "–". At the decoder, map a "0" received level into a "zero", and both "+" and "–" as "one".

**Exercise 19-4.** Show that the following alternative description of the AMI coder is valid. The coder keeps track of the RDS $s_k$, which can only assume the values "0" and "+1", depending on

whether the last non-zero pulse had negative or positive polarity respectively (this assumes that the first non-zero symbol was "+"). The coder then obeys the following truth table:

| $b_k$ | $s_{k-1}$ | $a_k$ | $s_k$ |
|---|---|---|---|
| 0 | | 0 | $s_{k-1}$ |
| 1 | 0 | + | 1 |
| 1 | 1 | – | 0 |

In this table the blank entry for $s_{k-1}$ when $b_k = 0$ means that we don't care what $s_{k-1}$ is, and a "0" level is always transmitted.

A different but equally useful generalization of AMI follows from another description. We can think of the AMI code as providing two mappings from $b_k$ to $a_k$ depending on the RDS at the last symbol as in the following table:

| $s_{k-1} = 0$ | | $s_{k-1} = 1$ | |
|---|---|---|---|
| $b_k$ | $a_k$ | $b_k$ | $a_k$ |
| 0 | 0 | 0 | 0 |
| 1 | + | 1 | – |

The two mappings have the characteristic of being one-to-one (so that we can recover the data symbol at the receiver); one increases the RDS, while the other decreases it. We decide which mapping to use on the basis of the RDS at the last symbol, and in particular choose it to keep the RDS in the range [0, 1]. A special class of line coders, called *sequence-state coders* generalize this idea (Section 19.2.2).

Calculating the power spectrum of the AMI code represents an interesting challenge, as we need to use the Markov chain results of Chapter 3. Interestingly, the AMI precoder is the same as the parity check circuit used as an example in Chapter 3.3 (see Fig. 3-7). Let the input bit stream consist of independent bits, and let the probability of a "one" be $p$. We will find, not surprisingly, that $p$ will have a large influence on the power spectrum, which is determined for the precoder in Problem 3-29. The AMI coder output is the precoder output filtered by the linear time-invariant filter in Fig. 19-6,

$$H(z) = 1 - z^{-1} . \tag{19.18}$$

The output power spectrum is therefore obtained by multiplying by $H(z)H(z^{-1})$,

$$S_A(z) = \frac{p(1-p)(1-z^{-1})(1-z)}{(1-(1-2p)z^{-1})(1-(1-2p)z)} . \tag{19.19}$$

Evaluating on the unit circle we get the power spectrum

$$S_A(e^{j2\pi fT}) = 2p(1-p)\ \frac{1-\cos(2\pi fT)}{1+(1-2p)^2-2(1-2p)\cos(2\pi fT)} . \tag{19.20}$$

The power spectrum after pulse shaping can of course be found using (19.4). The power spectrum of (19.20) is plotted in Fig. 19-7. Note the nulls in the spectrum at $f = 0$, as expected, and also at all multiples of the symbol rate $f = 1/T$, due to the periodicity of the spectrum. Also note the influence of $p$, with large $p$ resulting in a preponderance of power near half the symbol rate, due to the alternating "+" and "–" pulses. The number of transitions in the coded waveform is directly related to the density of "ones" in the information bit stream; for ease of

timing recovery most systems that use AMI also place restrictions on the minimum density of "ones".

AMI shares with the twinned binary code one serious flaw: it is possible to have a long series of transmitted "0" levels, corresponding to a long series of "zero" inputs. This corresponds to a long period where the transmitted signal is zero, which can cause timing recovery circuits to lose synchronization (Chapter 16). This flaw was acceptable in the early days of PCM because the bit stream was always an encoded version of an analog signal, rather than a direct transmission of digital data.

**Example 19-5.** The T1 transmission system requires that the input bit stream have at least a single "one" out of every eight bits and no more than 15 "zeros" in a row. This requirement is usually met by insuring that the all-zero octet of eight bits is never transmitted. When the bit stream is an encoded analog signal, the possibility of long strings of "zeros" is precluded by simply ruling out the all "zero" quantizer level (with a slight increase in quantizer error).

In retrospect, the choice of AMI precluded direct transmission of user-provided data without an intermediate coding step to eliminate the all-zero sequence. More recent digital transmission systems have adopted one of several modifications to AMI, described in the problems, to circumvent this shortcoming.

## 19.2.2. Block Line Codes

AMI is an example of a line code that operates continuously on the input bit stream to generate a stream of data symbols. In a *block code*, the input stream is divided into blocks, each of which is translated into a block of data symbols. Assume a block of $k$ bits is mapped into a block of $n$ data symbols drawn from an alphabet of size $L$, with the constraint

$$2^k \le L^n \,. \tag{19.21}$$

When equality is not met in (19.21), redundancy is available that can be used to accomplish desirable goals such as minimizing baseline wander or providing energy for timing.

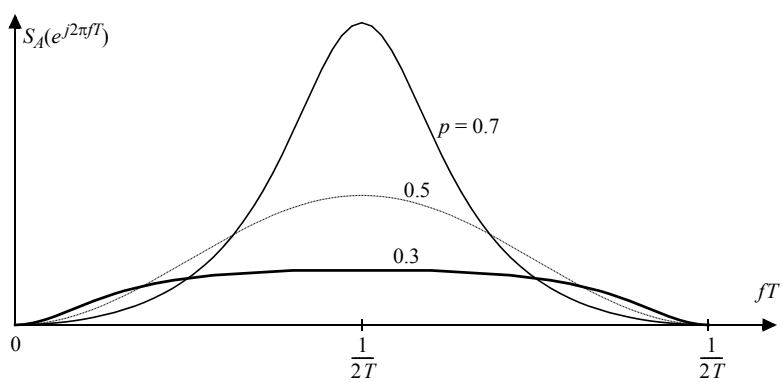



**Fig. 19-7.** Power spectrum for the AMI encoder output, neglecting the effect of the transmit pulse $g(t)$, for different density of "ones" in the information bit stream.

**B6ZS and HDBk**

The AMI code is so commercially important that a number of codes have been invented that fix the major problem with this code — the possibility of transmitting a signal with no timing energy for an arbitrary period of time. This problem is fundamentally due to the linearity of the AMI code, since linearity implies that the all-zero bit sequence is translated into a transmitted all-zero signal. The solution to this problem is to modify the line code to make it nonlinear. This class of codes modifies AMI by performing a substitution for a block of $k$ consecutive "0"'s that would otherwise be transmitted. The substituted block, which contains one or more non-zero symbols to ensure timing energy, uses the fact that only $2^{k+1}$ patterns of $k$ transmitted symbols ($2^k$ for each of the two values of the RDS at the beginning of the block) are allowed by AMI; one of the non-allowed blocks is substituted for the all-zero block. At the receiver, this non-allowed block can be recognized and replaced with the all "zero" decoded block.

This approach complicates the coder and decoder slightly, reduces the in-service monitoring capability of the code slightly, and increases the RDS. In spite of these disadvantages, these codes are widely used because of the critical need for reliable timing recovery. Two examples of these codes are considered in the problems — B6ZS in Problem 19-12 and HDBk in Problem 19-13.

**kBnT Codes**

AMI and its derivative pseudoternary codes transmit only one bit per symbol, whereas the capacity of a ternary symbol is $\log_2 3 = 1.58$ bits. In addition, AMI gives little control over the power spectrum. A much broader class of pseudoternary codes with the designation "kBnT" address these shortcomings, where $k$ is again the number of information bits and $n$ is the number of ternary symbols per block. If we choose the largest $k$ possible for each $n$, we get a table of possible codes (up to $k = 7$):

| $k$ | $n$ | Code | Efficiency |
|---|---|---|---|
| 1 | 1 | 1B1T | 63% |
| 3 | 2 | 3B2T | 95% |
| 4 | 3 | 4B3T | 84% |
| 6 | 4 | 6B4T | 95% |
| 7 | 5 | 7B5T | 89% |

In this table, we define the *efficiency* of the code as the ratio of the *rate* of the code in bits per symbol to $\log_2 3$. AMI is an example of a 1B1T code. As the block size increases we can generally achieve greater efficiency, but not without cost. Greater efficiency implies better noise immunity on many channels, since it translates into a lower symbol rate for a given bit rate and hence a reduced noise bandwidth. However, greater efficiency also implies reduced redundancy, and hence less control over the statistics of the transmitted signal (power spectrum, timing recovery, density of ones, etc.). The 4B3T code seems to be a reasonable compromise between these competing goals, and has been widely studied and used in some digital subscriber loop applications [9].

A complication in kBnT line codes is the need for the decoder to know the boundaries of the blocks of $n$ ternary symbols. This would normally be accomplished by framing, which is required in multiple channel systems in any case.

**Table 19-1.** An example of a 4B3T code illustrating a bimode block line code.

| Input Block | Ternary Output Block | | Block |
|---|---|---|---|
| | Mode A | Mode B | Digital Sum |
| 0000 | + 0 – | + 0 – | 0 |
| 0001 | – + 0 | – + 0 | 0 |
| 0010 | 0 – + | 0 – + | 0 |
| 0011 | + – 0 | + – 0 | 0 |
| 0100 | + + 0 | – – 0 | ±2 |
| 0101 | 0 + + | 0 – – | ±2 |
| 0110 | + 0 + | – 0 – | ±2 |
| 0111 | + + + | – + + | ±3 |
| 1000 | + + – | – – + | ±1 |
| 1001 | – + + | + – – | ±1 |
| 1010 | + – + | – + – | ±1 |
| 1011 | + 0 0 | – 0 0 | ±1 |
| 1100 | 0 + 0 | 0 – 0 | ±1 |
| 1101 | 0 0 + | 0 0 – | ±1 |
| 1110 | 0 + – | 0 + – | 0 |
| 1111 | – 0 + | – 0 + | 0 |

In designing kBnT codes, we must first recognize that we cannot obtain reasonable efficiency with zero-disparity code words; that is, code words with a digital sum of zero. For example, with three ternary digits, there are only six ternary code words with zero-disparity (assuming no use of the all-zero codeword, which would lead to long strings of zeros as in AMI). But as with AMI, we may define a set of different mappings (called *modes*) between binary words and ternary words, with the use of feedback to choose the appropriate mode at each block so as to minimize the RDS. Such codes are known as a *sequence-state line codes*; the first such code was described by Franaszek in 1968 [1]. The simplest such code has two modes, like AMI, although many codes have been proposed with three and four modes. A *bimode* (two-mode) 4B3T code is illustrated in Table 19-1.The remaining specification of the code is that Mode A is chosen whenever the RDS at the beginning of the block is in the range $3 \leq RDS \leq -1$ and Mode B is chosen when $0 \leq RDS \leq 2$.

Upon examination, the design principles for this code are straightforward. There are seven zero-disparity blocks of three ternary symbols; six of these are assigned to six input blocks for both modes. These can be assigned to both modes since they do not affect the RDS at the end of the block. The seventh zero-disparity ternary block, "000," is not used because it has no timing energy. The remaining $27 - 7 = 20$ ternary code words are assigned to the remaining 10 input blocks, the positive disparity blocks to Mode A and the negative disparity blocks to Mode B. Whenever the RDS is positive, Mode B is used to make the RDS smaller, and conversely Mode A is used to make the RDS larger when it is negative. The largest change in the RDS over one block is three. The RDS at the ends of the blocks is in the range $-3 \leq RDS \leq 2$. Examination reveals that the RDS can increase by one after the first ternary symbol in the block, so that within the block the RDS is bounded by $-4 \leq RDS \leq 3$, and the DSV is seven. Hence, we pay a fairly substantial price in RDS for the greater efficiency of the 4B3T as compared to the 1B1T (AMI) code.

The decoder for 4B3T simply slices the ternary data, and does a table lookup to determine the binary block. Note that this is a memoryless function, independent of the state of the encoder or the RDS. Hence, there is no mechanism for error propagation. This memoryless decoder throws away all information about the sequence of states at the encoder; this additional information could be exploited by a sequence-state ML detector (the Viterbi algorithm of Chapter 6) to reduce the error rate.

**Binary Block Codes**

A special case occurs when $L = 2$ in (19.21), which means that the signal is binary. For this case, we have the simpler constraint that

$$k \le n ; \tag{19.22}$$

in other words, the block of $n$ binary data symbols must be longer than the number $k$ of information bits at the input to the line coder. Binary block codes are useful for media that are not well-suited to transmitting other than two-level signals, or when the additional bandwidth required for a binary transmitted signal is easier to achieve than increasing the number of levels.

**Example 19-6.** In optical fibers (Section 18.3), intensity modulation is usually used, so that the information content is transmitted as signal intensity or power. Hence, it is possible to transmit only zero and positive levels, not negative levels. Furthermore, a modest increase in the symbol rate usually costs little in terms of faster electronics, noise immunity, or achievable distance between repeaters.

**Example 19-7.** In magnetic recording (Section 18.6), the medium is highly nonlinear unless a.c. bias recording is used. It makes sense to operate the medium in a binary saturation mode, with one of two magnetic polarizations. Further, it is possible to achieve a high effective information rate by encoding the information by the location of transitions in the waveform. The bandwidth of the waveform is therefore less important than the accuracy with which the location of a transition can be generated and detected. Increasing this accuracy, and hence the effective symbol rate, costs little in terms of noise immunity.

One primary motivation for the design of line codes has been the elimination of the d.c. content of the coded signal because of a.c. coupling to the medium. It might appear that this problem does not occur for media such as optical fiber and magnetic recording, since transformers are not required. However, it is difficult to build d.c.-coupled high-speed electronics for preamplification, etc. Furthermore, the design can often be simplified by choosing as high a cutoff frequency as possible.

Binary block codes with no d.c. content can be designed by maintaining a balance between the number of positive and negative transmitted symbols in each codeword. This is usually a key objective in the design of the line code.

We can choose either a *zero-disparity code* or a *bimode code*. In a zero-disparity code [10][11], each block of $n$ transmitted bits is constrained to have $n/2$ "ones" and $n/2$ zero bits, thus maintaining an $RDS = 0$ at the end of each block. Obviously $n$ must be an even number. The number of possible code words is precisely

$$N = \frac{n!}{(n/2)!(n/2)!} \tag{19.23}$$

in accordance with the number of possibilities for locating $n/2$ "ones" in $n$ positions. The number of available code words and information capacity are tabulated in the following table:

| $n$ | $N$ | $\log_2 N$ | $k$ | Efficiency |
|---|---|---|---|---|
| 2 | 2 | 1 | 1 | 63% |
| 4 | 6 | 2.58 | 2 | 95% |
| 6 | 20 | 4.32 | 4 | 84% |
| 8 | 70 | 6.13 | 6 | 95% |
| 10 | 252 | 7.97 | 7 | 89% |

The efficiency generally increases with block size, although not monotonically due to the constraint that the number of codewords be a power of two.

**Example 19-8.** With $n = 10$, we are tantalizingly close to eight information bits, but unfortunately we must settle for seven. The efficiency is therefore 70%. Of course, we can take advantage of this additional redundancy by choosing only the 128 out of the 252 zero-disparity codewords that have the most desirable properties. For example, we might choose codewords that have the highest timing content, or those that maintain the smallest RDS within the codeword.

The RDS of a zero-disparity code is generally limited to the range $-n/2 \le RDS \le n/2$, although a smaller range can be achieved with a larger redundancy and lower efficiency.

A higher efficiency can be obtained by using a bimode code. In this case we include some codewords with non-zero but small disparity. We group the blocks into two modes: a positive mode, containing all blocks with positive disparity, and a negative mode, containing all blocks with negative disparity. Blocks with zero disparity can be included in both modes. The line coder selects, for each block, the mode that will reduce the magnitude of the RDS. As in the pseudoternary block codes, the code is constructed to ensure that the decoding is memoryless, independent of the state of the coder, so that there is no error propagation mechanism.

While we have emphasized the use of line coding to ensure zero d.c. content in the signal, a code can also be used to tailor the signal to other properties of the channel.

**Example 19-9.** A $(d, k)$ code is often used in magnetic recording. In a $(d, k)$ code, the coded data bits meet the constraint that the number of consecutive zeros must be at least $d$ and at most $k$. The IBM 3380 disk magnetic storage system uses a (2, 7) code; i.e., runs of zeros are always at least two and at most seven in length [12]. The magnetic medium is characterized by the maximum allowable flux changes per inch (FCI) along the recording track; the binary data symbols are encoded as flux changes. With $(d, k)$ codes with $d > 0$, it is possible to achieve more binary data symbols per inch than the FCI. We can write symbols at a density of $FCI \cdot (d + 1)$ binary symbols per inch without violating the condition that there be no more than FCI flux changes per inch. In effect we are transmitting faster than the FCI by encoding the information as the interval between transitions, and increasing the resolution while maintaining the FCI constraint. The upper bound $k$ on the number of consecutive zeros is dictated by timing recovery considerations, since a zero is encoded as no flux change.

For a $(d, k)$ code define $C(d, k)$ as the maximum number of information bits that can be achieved per coded binary symbol. Of course

$$C(d, k) \le C(0, \infty) = 1 . \tag{19.24}$$

Then the increase in the recording density is $(d + 1) \cdot C(d, k)$. The calculation of the capacity is rather complicated [12][13].

### 19.2.3. Variable-Rate Codes

Some media and multiplexing methods allow the transmitted symbol rate to be variable. The philosophy is to meet the constraints of the channel and at the same time minimize the *average* transmitted symbol rate.

**Example 19-10.** In a magnetic or optical recording system, the number of recorded symbols per information bit need not be predictable, but we would like to minimize the average total number of symbols recorded.

**Example 19-11.** When *statistical multiplexing* techniques are used (Chapter 17), users' messages are interleaved with stuffing information to fill out a fixed-rate bit stream. The length of a user's message is unpredictable, and after coding the number of transmitted symbols need not be predictable.

As a simple but important practical example of a variable-rate coding scheme, consider *bit-stuffing* to meet a $(0, k)$ run-length constraint. Here the objective is to ensure that no more than $k$ consecutive zeros occur in a coded binary sequence (usually to meet timing recovery constraints). The technique is simple and effective — simply *add* or *stuff* an extra "one" after every $k$ "zeros." The decoder simply removes the obligatory "one" at the end of every $k$ consecutive "zeros."

**Example 19-12.** A $(0, 2)$ run-length limited binary code would encode the sequence "100011001" as "10010110011." The decoder simply replaces every sequence "001" by "00" to recover the original bit sequence. Note that the fragment "001" was mapped into "0011;" the addition of the stuffed "1" would seem to be unnecessary because the original sequence met the run-length constraints. However, the stuffed bit is necessary for correct decoding. The coded sequence is longer than the uncoded sequence (eleven symbols vs. nine bits). The length of the coded sequence is dependent on the input information bits, and hence the code has variable rate.

The number of coded symbols can be predicted only statistically from the statistics of the information bits.

**Exercise 19-5.** Assuming the information bits are independent and identically distributed with $q$ the probability of a "zero," show that the coded sequence has average bit rate

$$1 + \frac{(1-q)q^k}{1-q^k} \approx 1 + (1-q)q^k \tag{19.25}$$

times the input bit rate. For example, for $q = 1/2$ the overhead is approximately a fraction $(1/2)^{k+1}$ which can be very small for large $k$. (*Hint:* Use the results of Problem 3-13.)

## 19.3. FILTERING FOR SPECTRUM CONTROL

In Section 19.2.1, several linear line codes for introducing spectral nulls were described. For example, in the twinned binary code of Fig. 19-3, the transmitted symbols were filtered by a transfer function $(1 - D)$. The result was a new sequence of pseudoternary symbols (three levels, but only one bit of information) with a null in the power spectrum at d.c. This approach can be generalized by passing the transmitted symbols through an arbitrary filter. However, so far, no systematic design methodology has been introduced for designing such codes. This

shortcoming will be rectified in this section, where the precoding technique will be generalized. The idea is very simple: spectral nulls can be introduced by putting an appropriate filter into the transmitter. That filter introduces ISI, but the ISI can be eliminated by precoding in the transmitter, at the expense of an expanded slicer in the receiver. In the case of a transmit filter with integer coefficients, we obtain an important practical technique known as *partial response*. AMI coding (Section 19.2.1) is a special case of partial response.

### 19.3.1. Adding Spectral Nulls Using Precoding

Spectral nulls can be inserted at any frequency or finite set of frequencies using a filter in the transmitter. As pointed out in Section 8.1.4, transmit filtering often has the undesirable side effect of increasing the peak transmitted power. In addition, ISI is introduced. AMI coding (Section 19.2) is an example of an approach that achieves a spectral null without ISI, at the expense of increasing the number of levels in the slicer, and a modest increase in peak and average transmitted power (for the same minimum distance).

We will now show that AMI can be generalized, using the precoding technique defined in Section 8.1.4. Recall that the purpose of precoding is to combat ISI in the transmitter, using nonlinear modulo arithmetic to avoid large increases in peak transmitted power. The precoder approach to generating spectral nulls in the transmitted signal is to put a null-generating filter in the transmitter, and then combat the resulting ISI using precoding, also in the transmitter. One of the disadvantages of precoding, the need to know the channel response accurately, is not a problem here because the filtering that introduces ISI is always accurately known to the precoder.

The basic approach, introduced in [14], is illustrated in Fig. 19-8. The output of a transmitter precoder is passed through a linear filter $F(z)$ that is monic, causal, and loosely minimum-phase. Using (2.51), $F(z)$ can be decomposed as

$$F(z) = F_{\min}(z)F_{\text{zero}}(z)\,, \tag{19.26}$$

where $F_{\min}(z)$ is a monic strictly minimum-phase filter and $F_{\text{zero}}(z)$ is a monic causal FIR filter with zeros on the unit circle. (In this subsection, we use $z^{-1}$ in place of $D$, because this is easier to relate to the results of Chapter 2.) Since we care only about the power spectrum of the transmitted symbols $Y_k$, specializing $F(z)$ to a minimum-phase filter does not limit our ability to control the power spectrum. The reason for a monic causal $F(z)$ ($F(\infty) = 1$) is that the precoder of Section 8.1.4 requires this property. Since our primary motivation is to introduce spectral nulls in the transmitted spectrum, $F_{\text{zero}}(z)$ is the essential ingredient. A reason for choosing $F_{\min}(z) \neq 1$ will appear shortly.

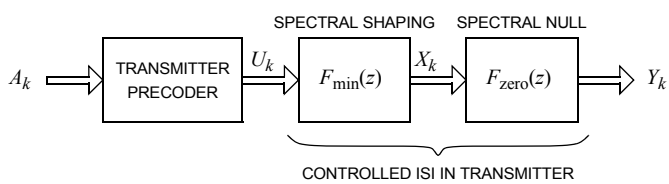



**Fig. 19-8.** Introducing spectral nulls in the transmitter using transmitter precoding. The data symbols are $A_k$, and the transmitted precoded symbols are $Y_k$.

If the channel is ideal, the Tomlinson precoder can be designed for the "channel" response $F(z)$ (actually a part of the transmitter). If the channel has a non-ideal response $H(z)$, where $H(z)$ is monic and causal, then the precoder can be designed for the "channel" response $F(z)H(z)$, which is also monic and causal. In either case, the receiver slicer must be replaced by an extended slicer.

Let $X_k$ be the output of $F_{\min}(z)$, as shown in Fig. 19-8. An important property of $X_k$ and $Y_k$ is that they are bounded, since the output of the precoder $U_k$ is bounded (because of the modulo operation in the precoder), and the filters are BIBO stable. (Because $F_{\min}(z)$ and $F(z)$ are both minimum-phase, they have no poles on or outside the unit circle, and hence they are BIBO stable (Chapter 2).)

Furthermore, using the continuous approximation for the input symbols $A_k$ (recall from Section 8.1.4 that this assumes they are i.i.d. and have a continuous uniform distribution), we saw in Section 8.1.4 that the $U_k$ are also i.i.d. (white) and uniformly distributed. With this approximation, the power spectra of $X_k$ and $Y_k$ are

$$S_X(e^{j\theta}) = \sigma_U^2 \, |F_{\min}(e^{j\theta})|^2 , \qquad S_Y(e^{j\theta}) = \sigma_U^2 \, |F(e^{j\theta})|^2 , \tag{19.27}$$

where $\sigma_U^2$ is the variance of $U_k$, which is approximately $M^2/3$ according to the continuous approximation. The continuous approximation requires that the constellation be square, and $M^2$ is the number of points in the constellation.

The motivation for including $F_{\min}(z)$ becomes apparent when we consider the case of a first-order spectral null at d.c.

**Example 19-13.** If we let $F_{\text{zero}}(z) = 1 - z^{-1}$, a spectral null at $z = 1$ is introduced. An important observation is that the input $X_k$ to the filter $(1 - z^{-1})$ is the running digital sum (RDS) of the output $Y_k$. We can see this from the relation

$$\sum_{m=0}^{k} Y_m = X_k - X_{-1} , \tag{19.28}$$

where we can assume $X_{-1} = 0$. Since $X_k$ is bounded, the RDS is bounded. Beyond this, it is advantageous to keep the RDS $X_k$ as small as possible, for the reasons explained in Section 19.1.

The purpose of the filter $F_{\min}(z)$ is to allow us to trade off $\sigma_X^2$ against $\sigma_Y^2$. Considering the first order null of Example 19-13, it is desirable to keep $\sigma_X^2$ small because it is a measure of the size of the RDS. We want to keep $\sigma_Y^2$ small, because it is directly proportional to the transmitted power. As we will see, there is a direct tradeoff between these goals, in that decreasing $\sigma_X^2$ results directly in an increase in $\sigma_Y^2$, and *vice versa*. Thus, there is a tradeoff between transmitted power and $\sigma_X^2$, and $F_{\min}(z)$ directly controls that tradeoff.

**Example 19-14.** Continuing Example 19-13,

$$\sigma_Y^2 = \mathrm{E}[|X_k - X_{k-1}|^2] = \sigma_X^2 \, (2 - \rho) , \qquad \rho = 2\mathrm{Re}\{\mathrm{E}[X_k X_{k-1}^*]\}/\sigma_X^2 \quad . \tag{19.29}$$

Thus, $\sigma_Y^2$ depends not only on $\sigma_X^2$, but also on $\rho$, which is related to the shape of $S_X(e^{j\theta})$, which is in turn controlled by $F_{\min}(z)$. Thus, for a given $\sigma_X^2$, the transmitted power can be influenced by the choice of $F_{\min}(z)$ through $\rho$. Clearly we want to choose $F_{\min}(z)$ to minimize $\sigma_Y^2$.

Since $U_k$ is white (by the continuous approximation) and $F_{\min}(z)$ and $F(z)$ are both monic, it follows that $\sigma_X^2 \geq \sigma_U^2$ and $\sigma_Y^2 \geq \sigma_U^2$. $F_{\text{zero}}(z)$ is chosen in accordance with the desired location and order of the spectral nulls. The remaining design problem is then stated as follows; Minimize $\sigma_Y^2$ subject to the constraint that $\sigma_X^2$ is fixed. This design problem has a feasible solution as long as we choose $\sigma_X^2 \geq \sigma_U^2$, and as we allow $\sigma_X^2$ to increase, the minimum $\sigma_Y^2$ will decrease. The solution to this problem is easy at the two endpoints. Choosing $F_{\min}(z) = 1$ allows $\sigma_X^2 = \sigma_U^2$, and results in the largest $\sigma_Y^2$ we have to accept ($\sigma_Y^2 = 2\sigma_U^2$ in the case of Example 19-13). At the other extreme, we can get an $\sigma_Y^2$ close to $\sigma_U^2$ by choosing $F_{\min}(z) \approx F_{\text{zero}}^{-1}(z)$, but doing so makes $\sigma_X^2$ very large since $F_{\text{zero}}^{-1}(z)$ is an unstable filter with poles on the unit circle. Thus, we can asymptotically force $\sigma_Y^2 \to \sigma_U^2$ (its minimum), but only by allowing $\sigma_X^2 \to \infty$.

Following [2][14][15], the optimal $F_{\min}(z)$ is easily found, based on the optimal linear predictor theory of Section 3.2.4. The constraint that $\sigma_X^2$ be held constant is

$$\sigma_X^2 = \sigma_U^2 \frac{1}{2\pi}\int_{-\pi}^{\pi} |F_{\min}(e^{j\theta})|^2 \, d\theta \ . \tag{19.30}$$

$F_{\min}(z)$ should be chosen to minimize

$$\sigma_Y^2 = \sigma_U^2 \frac{1}{2\pi}\int_{-\pi}^{\pi} |F_{\min}(e^{j\theta})|^2 |F_{\text{zero}}(e^{j\theta})|^2 \, d\theta \ . \tag{19.31}$$

This minimization can be solved using a Lagrange multiplier $\lambda \geq 0$ by performing the unconstrained minimization of

$$\frac{\sigma_Y^2 + \lambda\sigma_X^2}{\sigma_U^2} = \frac{1}{2\pi}\int_{-\pi}^{\pi} (|F_{\text{zero}}(e^{j\theta})|^2 + \lambda)|F_{\min}(e^{j\theta})|^2 \, d\theta \tag{19.32}$$

and then choosing $\lambda$ to satisfy the constraint on $\sigma_X^2$. Minimizing (19.32) with respect to a monic minimum-phase causal $F_{\min}(z)$ is precisely the problem of finding an optimal linear prediction error filter $F_{\min}(z)$ for a random process with power spectrum $(|F_{\text{zero}}(z)|^2 + \lambda)$, which is a spectral factorization problem (Section 3.2.4). Writing the minimum-phase spectral factorization of this spectrum,

$$|F_{\text{zero}}(z)|^2 + \lambda = A_G^2 \cdot G(z)G^*(1/z^*) \, , \tag{19.33}$$

where $G(z)$ is a minimum-phase transfer function, $\sigma_Y^2$ is minimized by choosing $F_{\min}(z) = 1/G(z)$. Since $(|F_{\text{zero}}(z)|^2 + \lambda)$ is always an all-zero FIR filter, $G(z)$ will always be an all-zero filter of the same order. Thus, the optimal $F_{\min}(z)$ is always an all-pole filter of the same order as $F_{\text{zero}}(z)$, and the optimal $F(z)$ is a filter with an equal number of zeros and poles, with the zeros on the unit circle and the poles inside the unit circle. Intuitively what the poles accomplish is to compensate for the gain introduced in $F_{\min}(z)$ at frequencies far away from its nulls, reducing the transmitted power. Simultaneously, they have the effect of narrowing the spectral null, which in some system contexts is an undesirable side effect, and of course also increasing $\sigma_X^2$, which is also undesirable.

**Example 19-15.** Continuing Example 19-13, since $F_{\text{zero}}(z)$ is first-order, $G(z)$ will also be first-order; namely, $G(z) = 1 - \beta z^{-1}$ for $|\beta| < 1$. The spectral factorization problem is thus

$$\cdot ((1 - z^{-1})(1 - z) + \lambda) = A_G{}^2 \cdot (1 - \beta z^{-1})(1 - \beta^* z) . \tag{19.34}$$

Equating the coefficient of $z^{-1}$, we see immediately that $\beta$ is real. With this information, we can abandon the parameter $\lambda$, and find $\sigma_X^2$ and $\sigma_Y^2$ as a function of $\beta$. Performing a partial fraction expansion of $S_X(z)$,

$$S_X(z) = \frac{\sigma_U^2}{(1 - \beta z^{-1})(1 - \beta z)} = \frac{\sigma_U^2}{1 - \beta^2}\left(\frac{\beta z^{-1}}{1 - \beta z^{-1}} + \frac{1}{1 - \beta z}\right) . \tag{19.35}$$

The variance $\sigma_X^2$ is the coefficient of $z^0$ in this expansion, which is

$$\sigma_X^2 = R_X(0) = \sigma_U^2 / (1 - \beta^2) . \tag{19.36}$$

It is also helpful to know the correlation of adjacent samples,

$$R_X(1) = \beta \sigma_U^2 / (1 - \beta^2) . \tag{19.37}$$

Finally, $\sigma_Y^2$ can be determined without a need to manipulate its power spectrum,

$$\sigma_Y^2 = 2(R_X(0) - \mathrm{Re}\{R_X(1)\}) = 2\sigma_U^2 / (1 + \beta) . \tag{19.38}$$

Note that $\beta = 0$, corresponding to no shaping filter, results in $\sigma_Y^2 = 2\sigma_U^2$, the worst-case transmit power. As we increase $\beta$, the pole location approaches the zero location, and $\sigma_Y^2$ decreases toward $\sigma_U^2$, its minimum value. However, simultaneously $\sigma_X^2 \to \infty$, because in effect $F_{\min}(z)$ is attempting to equalize $F_{\mathrm{zero}}(z)$, which is impossible. The optimal tradeoff between $\sigma_X^2$ and $\sigma_Y^2$ in dB is plotted in Fig. 19-9, where $\beta$ is a free parameter that is varied to trace the tradeoff. Any point on the curve can be achieved by the optimal filter design, any point below the curve cannot be achieved by any filter design; points above the curve can be achieved by suboptimal filter designs. The magnitude response of the filter $F(z)$ is shown in Fig. 19-10. As $\beta$ increases from zero, the width of the d.c. null decreases and the gain of the filter decreases at high frequencies (which is why the transmitted power decreases).

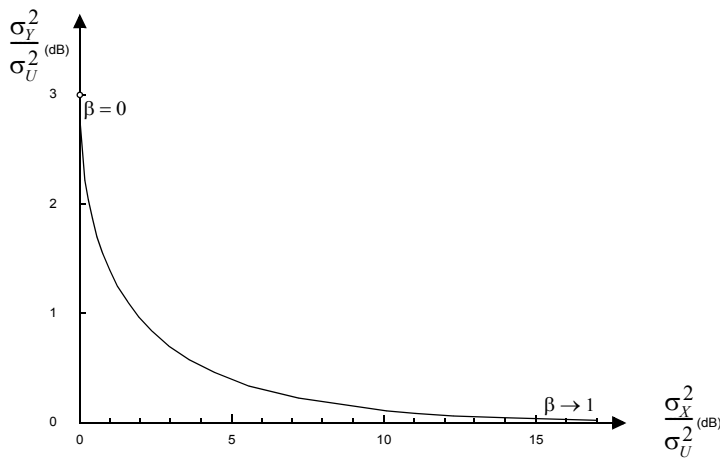



**Fig. 19-9.** The tradeoff between $\sigma_X{}^2$ and $\sigma_Y{}^2$ for an optimal filter design for a first-order null at d.c., $F_{\mathrm{zero}}(z) = 1 - z^{-1}$.

## 19.3.2. Partial Response

The transmit filter $F(z)$ just designed did not take into account implementation complexity. One of the implementation complications of the optimization procedure is that the transmitter precoder is not a finite state machine, except for very special filter designs [16], and even then the number of states in the transmitter may be very large. However, for some very simple choices of $F(z)$, the transmitter precoder becomes a finite-state machine with a small number of states that is simple to implement. AMI, it turns out, is an example. In particular, this happens when $F_{\min}(z) = 1$ and $F_{\text{zero}}(z)$ has integer-valued coefficients. The resulting system design is known as *partial response*, and has been used in many systems. The price paid for this simplicity is a larger transmitted power, relative to what can be obtained by choosing $F_{\min} \neq 1$, although desirably this case results in the smallest RDS.

Consider a filter response (we return to the notation $D = z^{-1}$ used in Section 19.2)

$$F(D) = \sum_{i=0}^{N} f_i D^i, \qquad f_0 = 1. \tag{19.39}$$

Three cases of particular interest are illustrated by the following three examples.

**Example 19-16.** The twinned binary line code used $F(D) = 1 - D$, introducing a single zero at d.c. ($D = 1$ or $f = 0$). The discrete-time frequency response of filter $F(D)$ is

$$F(e^{j2\pi fT}) = 1 - e^{j2\pi fT} = 2je^{-j\pi fT}\sin(\pi fT) \ , \tag{19.40}$$

is known as *dicode*, and is plotted in Fig. 19-11a.

**Example 19-17.** When $F(D) = (1+D)$, then a zero is introduced at half the symbol rate ($D = -1$ or $\omega = \pi/T$). The resulting frequency response,

$$F(e^{j2\pi fT}) = 1 + e^{j2\pi fT} = 2e^{-j\pi fT}\cos(\pi fT), \tag{19.41}$$

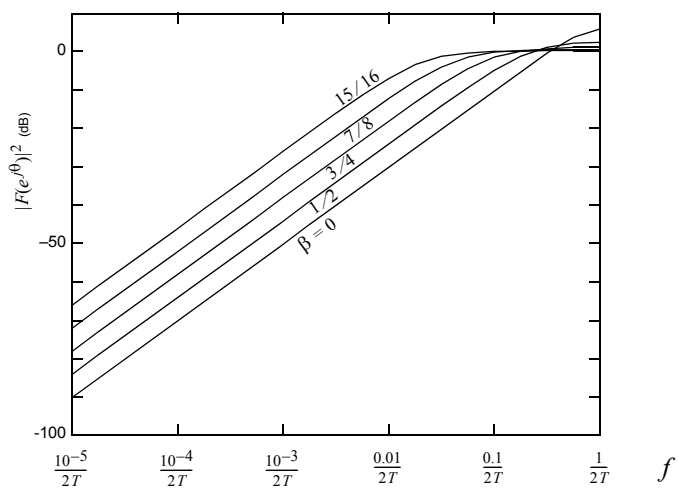



**Fig. 19-10.** The magnitude response of $F(z)$ plotted against frequency on a log scale. The left side of the frequency scale approaches d.c., with the right side is half the sampling rate. For $\beta = 0$, the gain at high frequencies results in a doubling of transmit power, and as $\beta \to 1$, this high frequency gain is reduced and the d.c. null is narrowed.

is known as *duobinary* and is plotted in Fig. 19-11b. This zero can be beneficial because it allows us to build practical digital communications systems with *no excess bandwidth*. Excess bandwidth is usually necessary because otherwise there would be a discontinuity in the spectrum of a transmitted pulse with an ideal lowpass characteristic. With a duobinary filter in the transmitter, it is possible to use zero excess bandwidth and still have no discontinuity in the spectrum.

**Example 19-18.** We can achieve zeros at both d.c. and half the symbol rate by choosing

$$F(D) = (1 - D)(1 + D) = 1 - D^2 . \tag{19.42}$$

The resulting frequency response,

$$F(e^{j2\pi fT}) = 1 - e^{-j4\pi fT} = 2je^{j2\pi fT}\sin(2\pi fT) , \tag{19.43}$$

is known as *modified duobinary* and is plotted in Fig. 19-11c. This case provides the advantages of both dicode and duobinary.

The filters in these examples place zeros in the spectrum at d.c. and/or half the symbol rate. More generally, a spectral shaping filter can be used to put an arbitrary number of zeros at these frequencies. For this case we can choose

$$F(D) = (1 - D)^m(1 + D)^n . \tag{19.44}$$

We allow $m = 0$ or $n = 0$, but not both. Filters in this form have the special properties that the coefficients are integer-valued and the transmitter precoder has a relatively small number of states. They do not exhaust the possibilities, but include the cases of practical interest.

While filtering the data symbols with $F(D)$ does have obvious advantages, what are the problems introduced? Two problems can be immediately recognized:

- For dicode, duobinary, and modified duobinary, if we put a binary antipodal data symbol $\{\pm 1\}$ into the filter, the output is three-level $\{\pm 2, 0\}$. Since there is only one bit of information conveyed per output symbol, the resulting code is pseudoternary. Choosing

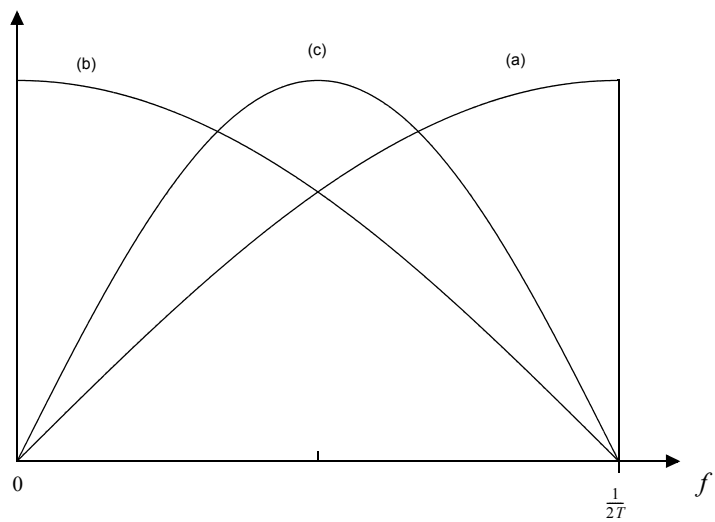



**Fig. 19-11.** The frequency response of $F(D)$ plotted up to half the symbol rate. (a) Dicode, (b) duobinary, and (c) modified duobinary.

other $F(D)$'s can result in a much larger constellation at the output (generally as $m$ and $n$ increase in (19.44), the size of the constellation increases). This expansion of the constellation size is the price paid for correlation of successive symbols and the resulting control over the spectrum. The larger constellation also potentially reduces noise immunity. For example, if we constrain the peak transmitted signal, the larger constellation with $F(D)$ will result in a smaller spacing between levels.

- The filter $F(D)$ introduces ISI. Since this has been done deliberately, rather than as a side effect of the channel, we call this *controlled ISI*.

Once we have introduced controlled ISI in order to affect the spectrum of the transmitted signal, we have several options for equalization of that ISI, as described in Chapter 8.

**Example 19-19.** Suppose we transmit a binary antipodal signal $C_k = \pm 1$ filtered by the dicode response $F(D) = 1 - D$ of Example 19-16, and the equivalent discrete-time channel introduces no intersymbol interference of its own but only additive noise $N_k$. Four options for detecting $C_k$ are shown in Fig. 19-12. In Fig. 19-12a we use a linear equalizer (LE-ZF, Chapter 8) $F^{-1}(D) = 1/(1 - D)$, which restores a binary signal $C_k$; hence a binary slicer can be used. Unfortunately, since $1/(1 - D) = 1 + D + D^2 + \ldots$, for independent noise samples the noise enhancement of this filter is infinite! More generally, a LE-ZF equalizer will suffer infinite noise enhancement for any filter of the form of (19.44) because of the algebraic zero in the frequency response. A second option, the decision-feedback equalizer of Fig. 19-12b, also uses a binary slicer, but eliminates the noise enhancement by canceling the ISI (which is postcursor) using the past decision. Unfortunately, it has error propagation. A third option exploits the ternary nature of the input signal $(C_k - C_{k-1})$ by applying it directly a ternary slicer in Fig. 19-12c. In the absence of noise ($N_k = 0$) this slicer has no effect on the signal; hence we can place the linear equalizer filter after the slicer. This option eliminates the noise enhancement of Fig. 19-12a, because the noise is removed by the slicer prior to equalization, but unfortunately also results in error propagation ( Problem 19-20). The fourth option of Fig. 19-12d is to use the Viterbi algorithm. This approach will have the best performance.

### Precoding

There is another way to deal with the ISI, illustrated in Fig. 19-13, which is a generalization of the approach we used to go from twinned binary to AMI in Section 19.2. We put the input bit stream through a *precoder* prior to the spectral shaping filter $F(D)$. In the receiver, we use a

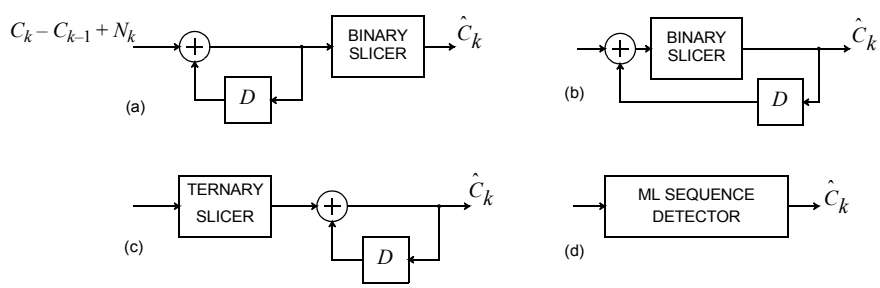



**Fig. 19-12.** Four options for detecting a dicode pseudoternary signal. a. A linear equalizer filter $F^{-1}(D)$. b. A decision-feedback filter with binary slicer. c. A filter $F^{-1}(D)$ after a ternary slicer. d. A ML sequence detector (Viterbi algorithm.)

configuration similar to Fig. 19-12c; namely, a slicer with more than two levels (three levels for the dicode case) followed by a *decoder* to recover the bit stream. Because of the precoder, in contrast to Fig. 19-12c, the decoder is *memoryless*, and there is *no error propagation*.

The combination of a precoder with spectral shaping filter $F(D)$ in the form (19.44) is called *partial response (PR)*. It is a simple technique for gaining the benefits of spectral shaping, with an implementation simpler than the ML sequence detector, and without the noise enhancement of the LE or the error propagation of the DFE. AMI is a special case of PR also known as dicode partial response. Other important examples of PR are duobinary partial response (Example 19-17) and modified duobinary partial response (Example 19-18). PR is used most often to avoid baseline wander (dicode and modified duobinary) and signal with the minimum bandwidth promised by the Nyquist criterion (duobinary and modified duobinary). Partial response is also sometimes called *correlative coding* because it introduces correlation among the transmitted data symbols, as reflected in the nulls in the spectrum. This correlation is achieved through redundancy, as in other line coding techniques. In PR, this redundancy is in the form of an increase in the size of the signal constellation over and above that required to accommodate the information bit rate.

All of the examples that we have given — dicode, duobinary, and modified duobinary PR — result in changing a binary signal into a three-level signal at the slicer. Other PR polynomials $F(D)$ can result in a larger output constellation. It is easy to see that if a binary antipodal signal is the input to the filter $F(D)$, the output constellation always consists of integer values, because of the integer coefficients of the filter. In addition, the larger the order of the filter, the larger the number of integers in the output constellation.

Duobinary PR was invented by Lender [17], and was generalized in [18]. Although it is a special case of the transmitter precoding derived in Section 8.1.4, we will derive the appropriate transmitter finite-state machine by a slightly different technique. We discuss precoding next, followed by the filter design and noise considerations.

We will now limit our attention to polynomials of the form (19.44), and design the appropriate precoder. The result is a systematic method for designing precoders, and leads to the same differential precoder design for dicode PR as was considered in Section 19.2 in the context of AMI. Assume that the input to the precoder is a bit stream, $b_k$, and the output is binary antipodal, $c_k = \pm 1$. It is important to note that in (19.44), $f_0 = F(0) = 1$.

We have already noted that the data symbols $a_k$ at the output of the filter $F(D)$ have a signal constellation that includes only integers because of the integer coefficients in the filter. In fact, because of the presence of a $(1 + D)$ or $(1 - D)$ factor, the output constellation includes only *even* integers, as we will show shortly. The implications are as follows. Two examples of signal constellations consisting of even integers are shown in Fig. 19-14, one with three points and one with five points. Recall that this constellation is redundant, and is used to communicate only one bit of information. Divide this constellation into two sets of points, one set $\Omega_0$

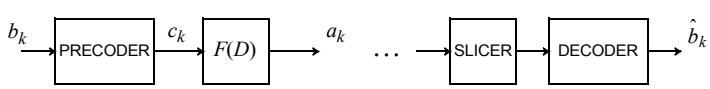



**Fig. 19-13.** Partial response adds precoding to $F(D)$ to allow detection without noise enhancement or error propagation.

corresponding to input $b_k = 0$, and the other set $\Omega_1$ corresponding to $b_k = 1$. Then the decoder can simply note whether the slicer output is in $\Omega_0$ or $\Omega_1$, and put out the appropriate bit. This slicer design is memoryless, as promised.

In fact, the points in the constellation should alternate between $\Omega_0$ and $\Omega_1$ as shown in Fig. 19-14 (where we have arbitrarily chosen $0 \in \Omega_0$). Why? Since $f_0 = 1$, the current $c_k$ at the precoder appears directly in the filter output. Thus, depending on $c_k$, the filter output is one of *two adjacent* even integers. The precoder therefore doesn't have complete control over $a_k$, but it can determine whether the output is in $\Omega_0$ or $\Omega_1$, and thereby communicate one bit of information. At the receiver, the slicer expects a constellation of even integers, and thus applies thresholds at odd integers (two thresholds for a three-level slicer, four for a five-level slicer).

We still need to show that when the filter $F(D)$ input is binary antipodal, the output is an even integer. Taking the case where $F(D)$ has a factor $(1 + D)$, or $n \geq 1$, we can write

$$F(D) = J(D) \cdot (1+D) , \qquad J(D) = (1 - D)^m (1 + D)^{n-1} , \tag{19.45}$$

where $J(D)$ has integer-valued coefficients. The D-transform of the filter output is

$$A(D) = F(D) \cdot C(D) = J(D)(1+D) \cdot C(D) . \tag{19.46}$$

The coefficients of $(1+D)C(D)$ are the sum of two coefficients of $C(D)$ and hence must be in the set $\{0,\pm 2\}$. Since $J(D)$ has integer-valued coefficients, the coefficients of $J(D)(1+D)C(D)$ must be even. A factor of $(1 - D)$ rather than $(1+D)$ in $F(D)$ would not change this result.

The design of the precoder is best illustrated by example.

**Example 19-20.** *Dicode partial response.* Let $F(D) = 1 - D$, and hence $a_k = c_k - c_{k-1}$. The constellation for $a_k$ includes the three levels $\{0, \pm 2\}$ shown in Fig. 19-14a. We can fill in the following precoding table:

| $c_{k-1}$ | $b_k$ | $c_k$ | $a_k$ |
|---|---|---|---|
| –1 | 0 | –1 | 0 |
| –1 | 1 | +1 | +2 |
| +1 | 0 | +1 | 0 |
| +1 | 1 | –1 | –2 |

How were the entries in this table determined? For example, if $c_k - 1 = -1$, then $a_k = c_{k+1}$, which assumes the values 0 or +2. By convention in Fig. 19-14 we have assigned these points in the constellation to input bit 0 and 1 respectively, and this determines both $a_k$ and $c_k$.

This precoding can be implemented by logic operations on the incoming data bits. If the precoder output levels $c_{k-1}$ and $c_k$ are represented by bits ( –1 is a logic "0", and +1 a logic "1"), then the

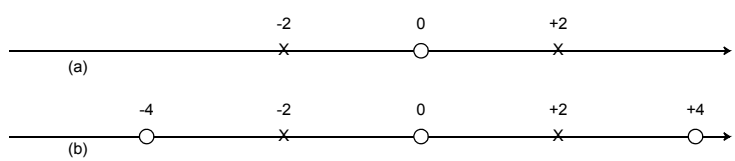



**Fig. 19-14.** A three-point (a) and five-point (b) signal constellation consisting of even integers. Points in set $\Omega_0$ are marked "o," and points in set $\Omega_1$ are marked "x.".

following modified table results:

| $c_{k-1}$ | $b_k$ | $c_k$ |
|---|---|---|
| 0 | 0 | 0 |
| 0 | 1 | 1 |
| 1 | 0 | 1 |
| 1 | 1 | 0 |

This table can be represented as an "exclusive-or" relation $c_k = c_{k-1} \oplus b_k$ as shown in Fig. 19-15. The precoder we have derived is identical to that for AMI given in Fig. 19-6. The decoder is also very simple — by convention a 0 level at the slicer is decoded as a "0", and the levels –2 and +2 are decoded as a "1".

**Example 19-21.** *Duobinary partial response.* For this case $F(D) = 1 + D$, and the following table is readily developed for the precoder:

| $c_{k-1}$ | $b_k$ | $c_k$ | $a_k$ |
|---|---|---|---|
| –1 | 0 | +1 | 0 |
| –1 | 1 | –1 | –2 |
| +1 | 0 | –1 | 0 |
| +1 | 1 | +1 | +2 |

For duobinary, the sense is the opposite of dicode: an input "1" results in no change in the precoder output, whereas an input "0" causes a reversal. In this case error propagation without precoding results because two input sequences, 010101… and 101010… result in the all-zero sequence after filtering. With precoding, only one input sequence — all zero — results in this filter input.

**Example 19-22.** *Modified duobinary partial response.* The PR polynomial $F(D) = 1 - D^2$ is equivalent to interleaving two dicode PR systems with polynomial $F(D) = 1 - D$. Therefore we use two independent dicode PR precoders, one operating on even-numbered input bits and the other on odd-numbered input bits. This is equivalent to a two-way interleaved AMI coder, as discussed in Section 19.2.

### Partitioning of Filtering in Partial Response

We have thus far defined $F(D)$ as the frequency response of a symbol-rate sampled filter that is a part of the transmitter. The purpose of this configuration would typically be to introduce nulls into the transmitted spectrum. In order to characterize the effect of PR on the signal spectrum, we must evaluate the effect of the precoder on the spectrum.

**Exercise 19-6.** Show that for dicode, duobinary, and modified duobinary PR, if the input bits to the precoder are independent and the probability of a "zero" is 1/2, then the precoder output bits are also independent. Thus, the spectrum at the output of $F(D)$ is affected only by the filter and not by the precoder. Note that this result is *not* valid if the input bits are not equally probable, in which case the precoder does modify the spectrum (Problem 19-22).

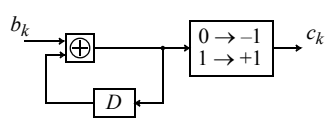



**Fig. 19-15.** The precoder for dicode partial response.

We must also consider the spectrum of a continuous-time PAM signal with symbol-rate sampled response $F(D)$. Define an equalized pulse shape $g(t)$ satisfying the Nyquist criterion, and then let an isolated pulse be

$$h(t) = \sum_{i=0}^{N} f_i\, g(t - iT)\ . \tag{19.47}$$

The samples of this pulse at the symbol rate are precisely the desired PR response,

$$h(kT) = \begin{cases} f_k\,, & k \in \{0, 1, \ldots, N\} \\ 0\,, & \text{otherwise} \end{cases}\ , \tag{19.48}$$

and the Fourier transform of this pulse is

$$H(f) = F(e^{j2\pi fT})G(f)\ , \tag{19.49}$$

where $F(e^{j2\pi fT})$ is plotted in Fig. 19-11 for three cases. In the case of dicode PR, which is the same as AMI, if we attempted to use 0% excess bandwidth there would be a discontinuity in the spectrum at half the symbol rate; therefore, we require some excess bandwidth for this case ($g(t)$ must have bandwidth greater than $1/(2T)$). For both duobinary and modified duobinary, however, it is practical to use 0% excess bandwidth because the spectrum will have no discontinuity. This is the origin of the term "duobinary;" a symbol rate double that of binary signaling is possible for a given bandwidth (actually, since 100% excess bandwidth is not required for binary antipodal signaling, the advantage is smaller than this).

With excess bandwidth, $g(t)$ is not unique and thus the PR isolated pulse is not unique. However, for 0% excess bandwidth, $G(f)$ is uniquely an ideal lowpass filter, and the PR responses are

$$\begin{aligned} &\text{Duobinary:} && h(t) = \operatorname{sinc}(\pi t/T) + \operatorname{sinc}(\pi t/T - \pi) \\ &\text{Modified duobinary:} && h(t) = \operatorname{sinc}(\pi t/T) + \operatorname{sinc}(\pi t/T - 2\pi)\ . \end{aligned} \tag{19.50}$$

These pulses are plotted in Fig. 19-16. The cancellation of the tails of the two "sinc" functions in (19.50) results in a well-behaved response in spite of zero excess bandwidth. Also note the width of the pulses, which obviously reflect their narrower bandwidth compared to Nyquist pulses (Chapter 5).

While we have emphasized the application of PR to controlling the transmitted spectrum, which results in putting the $F(D)$ response into the transmitter, there are other possible motivations for using PR, and reasons to make all or a portion of the filter $F(D)$ a part of the

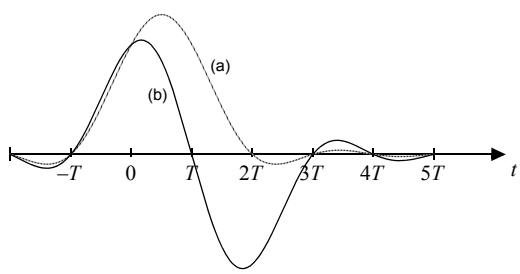



**Fig. 19-16.** Time response of equalized pulses for duobinary (a) and modified duobinary (b) with 0% excess bandwidth.

channel or the receiver. For example, the channel may naturally place a zero in the spectrum (for example, an a.c.-coupled channel will have a zero at d.c.), in which case a part of the $F(D)$ response may arise naturally. Alternatively, a zero placed in the receiver may help to reduce the noise at the slicer input, since it reduces the gain in the receive equalizer at some frequencies. In either of the latter cases, we can think of PR not as a method of spectrum control, but rather as an alternative to the DFE for improving noise immunity, but without the error propagation of the DFE.

Noise is an important consideration in the partitioning of filtering. It is often stated in the literature that there is a noise immunity penalty with PR. This is true on channels with little or no intersymbol interference, but is by no means generally true for other channels. We illustrate this by a simple calculation for two simple discrete-time channels. We will express the error probability in terms of the peak transmitted signal energy per symbol ($E_{\text{peak}}$) and the average transmitted signal energy per symbol ($E_{\text{avg}}$).

**Example 19-23.** *No intersymbol interference.* Let a channel be represented by the model

$$Y_k = X_k + N_k \tag{19.51}$$

where the zero-mean noise samples are independent with variance $\sigma^2$. Consider now four strategies:

*Binary antipodal signaling with two-level slicer:* For this case we let $X_k = \pm 1$ be a binary antipodal data symbol. The slicer input is $\pm 1$, and a noise sample must be larger than unity to cause an error. The error probability is therefore $Q(1/\sigma)$. The peak and average energies are the same, $E = 1$, and hence the error probability is $Q(\sqrt{E}/\sigma)$.

*Duobinary with transmit filtering to* $(1+D)$: For this case we let $X_k = C_k + C_{k-1}$, where $C_k$ is a duobinary precoded antipodal symbol, and the three-level slicer input $Y_k = 0, \pm 2$. Since the distance between slicer levels is two, the noise must again be larger than unity to cause an error, and the error probability will be a constant times $Q(1/\sigma)$. (The constant depends on end effects.) The peak transmitted energy (4) and average transmitted energy (2) are not the same (assuming independent input bits equally likely to be zero or one), so the error probability is $Q\left(\sqrt{0.25E_{\text{peak}}/\sigma}\right)$ or $Q\left(\sqrt{0.5E_{\text{avg}}/\sigma}\right)$. Duobinary is 3 dB poorer than binary antipodal with respect to average signal power, and 6 dB for peak signal power.

*Duobinary with receive filtering to* $(1 + D)$. For this case the transmitter is defined as $X_k = C_k$, the precoder output without the $(1 + D)$ filter, and the receiver applies $Y_k + Y_{k-1}$ to the three-level slicer. The variance of the noise at the slicer input is (because it is the sum of two independent noise samples) equal to $2\sigma^2$, and a noise larger than unity is required to cause an error. The peak and average energies are both unity, so the error probability is a constant times $Q(1/\sqrt{2}\,\sigma) = Q\left(\sqrt{0.5E/\sigma}\right)$, 3 dB worse than binary antipodal. This is because we have the same spacing of slicer levels, but twice the noise variance.

*Duobinary with transmit filtering and ML sequence detection.* For the transmit filtering case, the slicer input noise is white and we can apply ML sequence detection in place of simple slicing. The channel consists of binary antipodal symbols $C_k$ input to channel response $(1 + D)$, and as shown in Chapter 7, the minimum distance corresponds to a single error. In this case the error has magnitude two, so the minimum distance is $d_{\min}^2 = 8$. There are an infinite number of minimum-distance error events, each corresponding to alternating precoded symbols. After decoding, each of these error events results in only a *single* bit error. The error probability is thus upper bounded by $Q(\sqrt{8}/2\sigma)$, and since the peak and average energies are again 4 and 2 respectively, the error probability is

$Q\left(\sqrt{0.5E_{\text{peak}}/\sigma}\right)$ or $Q\left(\sqrt{E_{\text{avg}}/\sigma}\right)$. The performance is 3 dB better than for a simple slicer, and with respect to average transmitted energy is equal to binary antipodal.

This example illustrates why it is often stated that duobinary PR has a three dB noise penalty, as this is true on channels which lack intersymbol interference. In addition, it illustrates how the ML sequence detector can gain back all but a tiny bit of the penalty. (It is not strictly equivalent because we have ignored distances larger than the minimum distance.) The reason the ML detector has an advantage is that not all sequences of pseudoternary levels at the slicer are allowed; the ML detector uses this additional information to advantage. The example also shows how precoding is beneficial when the ML detector is used — it reduces the number of errors caused by the infinite set of minimum-distance error events.

Since PR is equivalent to a DFE with feedback filter $(F(D) - 1)$, but without error propagation, we know from Chapter 8 that on many channels with intersymbol interference a properly chosen PR can have a noise *advantage* over binary antipodal signaling, as opposed to the disadvantage displayed in the previous example. Consider the following simple case of a channel for which duobinary is advantageous.

**Example 19-24.** *Channel with duobinary ISI.* Let the channel of Example 19-23 be modified to

$$Y_k = X_k + X_{k-1} + N_k\,, \tag{19.52}$$

so the channel itself has the $(1 + D)$ duobinary response. For this case, in order to use binary antipodal signaling, we must first equalize the channel to eliminate the intersymbol interference in either the transmitter or receiver. Let us consider both transmitter and receiver equalization, along with duobinary with slicer and ML sequence detection.

*Binary antipodal with transmitter equalization:* Here we implement a filter $1/(1 + D)$ in the transmitter. Unfortunately, the peak transmitted signal is unbounded, since

$$\frac{1}{1+D} = 1 - D + D^2 - \ldots \tag{19.53}$$

and the transmitter output is a sum of an infinite number of binary antipodal symbols. Thus both the peak and average transmitted power are infinite.

*Binary antipodal with receiver equalization:* The LE-ZF equalizer puts a filter $1/(1+ D)$ in the receiver. Unfortunately the SNR is zero, since the noise at the slicer input

$$N_k - N_{k-1} + N_{k-2} - \ldots \tag{19.54}$$

now has infinite variance. The channel response with an algebraic zero cannot be equalized in the zero-forcing sense.

*Duobinary with no equalization:* Since the channel response is what we would like for duobinary, we can simply transmit $X_k = C_k$ where $C_k$ is the duobinary precoded signal, and apply the resulting channel output $Y_k$ directly to a ternary slicer. A noise larger than unity will cause an error, so the error probability is a constant times $Q(1/\sigma)$. Since both the peak and average transmitted powers are unity, the error probability becomes $Q(\sqrt{E}/\sigma)$.

*Duobinary precoding with ML sequence detection.* Again the minimum distance is $d_{\min}^2 = 8$, and hence the error probability is $Q(\sqrt{8}/2\sigma) = Q(\sqrt{2E}/\sigma)$. There is again a 3 dB advantage over the simple slicer.

This example illustrates the extreme case where the channel itself has a zero at half the symbol rate. We cannot use conventional Nyquist-pulse equalization since if we do the equalization in the transmitter, the peak signal becomes infinite, or if we do it in the receiver the noise becomes infinite. We must use a technique such as duobinary, or almost equivalently the DFE, and in the process we gain a large (infinite!) noise advantage. This example is extreme, since a channel with infinite loss at half the symbol rate is rare. However, channels with very large losses are common.

**Example 19-25.** Wire-pair and coaxial cable systems (Chapter 18.2) often operate at their maximum ranges with losses of 60 to 80 dB at half the symbol rate. For these channels, duobinary has a substantial noise advantage over binary antipodal signaling [19]. In fact, duobinary is used in wire-pair systems to achieve a doubling of bit rate (3 Mb/s vs. 1.5 Mb/s) over AMI (dicode PR) in the T1C transmission system.

Finally, we should reiterate that with PR precoding, although the number of levels is increased, not all sequences of data symbols are allowed (this is the redundancy). This fact can be used for performance monitoring (Problem 19-24) and, as with any channel with intersymbol interference, the Viterbi algorithm can be used to advantage in the decoding. Finally, PR can be generalized to an arbitrary number of transmitted levels ( Problem 19-25).

## 19.4. CONTINUOUS-PHASE MODULATION

*Continuous-phase modulation* (CPM) is a class of signaling schemes that maintains a constant envelope and avoids abrupt phase changes. The constant envelope is advantageous in many situations, particularly for channels with nonlinearities. Phase-shift keying with a rectangular pulse shape also maintains a constant envelope, but has phase discontinuities that result in a larger bandwidth for the transmitted signal.

A CPM signal is a phase modulated carrier,

$$X(t) = K\cos\left[2\pi f_c t + 2\pi h\int_{-\infty}^{t} S(\tau)d\tau + \phi\right], \tag{19.55}$$

where $h$ is called the *modulation index* and

$$S(t) = \sum_{m=-\infty}^{\infty} A_m\, g(t - mT) \ . \tag{19.56}$$

To maintain phase continuity, $S(t)$ must not have impulses.

**Example 19-26.** In Chapter 6 we considered special cases of CPM where the pulse $g(t)$ is rectangular, or constant over the interval 0 to $T$ and zero elsewhere. In this case, CPM is called *continuous phase FSK* (CPFSK). An interesting special case of CPFSK is MSK, as described in Chapter 6. The MSK signal of Chapter 6 signal can be written

$$X(t) = \cos\left[2\pi f_c t + \pi\int_{-\infty}^{t} S(\tau)d\tau + \frac{\pi}{2}\right], \tag{19.57}$$

where $S(t)$ is given by (19.56), $A_m = \pm 1$, and

$$g(t) = \frac{1}{2T}w(t) = \begin{cases} 1/(2T), & t \in [0, T) \\ 0, & \text{otherwise} \end{cases}. \tag{19.58}$$

The modulation index is $h = 1/2$.

**Exercise 19-7.** Show that (19.57) can be written in the form (6.117). Identify $b_m$ and $\phi_m$ in (6.117) and show that they satisfy (6.118).

It is common to normalize the pulse $g(t)$ (as in (19.58)) so that it integrates to $1/2$. With this normalization, the CPM signal has a phase change of $A_m h\pi$ radians in one symbol interval, with respect to the carrier $f_c$.

MSK provides a constant envelope signal with considerably narrower bandwidth than a constant envelope PSK (using rectangular pulses). However, since for CPFSK $g(t)$ in (19.56) is rectangular, there is a discontinuity in the first derivative of the CP signal. This implies that with smoother choices for $g(t)$ we can significantly reduce the bandwidth by ensuring continuous first, or even second or third, derivatives. The simplest case, called *full-response CPM*, uses a $g(t)$ that is zero outside the interval $0 \le t \le T$. The second case, called *partial-response CPM*, uses a $g(t)$ that extends over several symbol intervals. The term partial response, as in Section 19.3, refers to the deliberate introduction of ISI for spectrum control. In fact, the spectral properties of partial-response CPM signals are considerably better than full-response CPM and CPFSK [20].

The evolution of a CPM signal over time can be compactly described using a *phase diagram*. The phase diagram plots the phase term from (19.55)

$$2\pi h\int_{-\infty}^{t} S(\tau)d\tau \tag{19.59}$$

for all possible input symbols $A_m$.

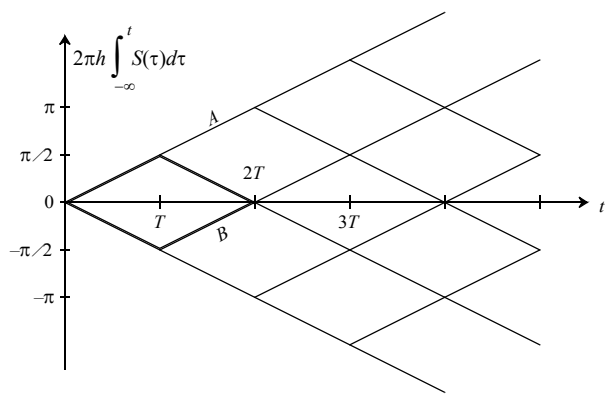



**Fig. 19-17.** A phase diagram for an MSK signal.

**Example 19-27.** The phase diagram for MSK is shown in Fig. 19-17.

Since phase is modulo $2\pi$, the phase diagram is best viewed wrapped around a cylinder with circumference $2\pi$. If the modulation index $h$ is rational, then the phase diagram will have a finite number of points on this cylinder. In this case, it can be viewed as the trellis diagram for a Markov chain. Since the phase evolves according to a Markov chain, the Viterbi algorithm is commonly used in the detector to perform optimal sequence detection.

**Example 19-28.** In Chapter 6 we found a receiver structure (Fig. 6-25) for MSK signals that performs roughly as well as the optimal matched-filter receiver for orthogonal FSK signals. However, by taking the absolute value of the sampled output of the matched filters, that structure discards useful information. The useful information that is discarded is easily seen in Fig. 19-17. The receiver in Fig. 6-25 makes no distinction between the path labeled "A" and the path labeled "B", which correspond to signals that are $\pi$ radians apart, or antipodal. Obviously, making the distinction would be useful. Considering that the phase is modulo $2\pi$, the phase diagram in Fig. 19-17 has only four states, best viewed as lying on a cylinder. The minimum-distance error event has length two and is the bold diamond shape in Fig. 19-17 (see Problem 19-26). Furthermore, the squared distance of this error event is twice the squared distance between the two orthogonal signaling pulses, so application of the Viterbi algorithm results in a 3 dB improvement over the receiver in Fig. 6-25. That receiver was shown to perform roughly as well as an optimal orthogonal FSK receiver, which is 3 dB worse than an optimal binary antipodal (2-PSK) signal of the same average power. Consequently, by using the Viterbi algorithm with MSK we are able to recover the 3 dB loss associated with FSK and match the performance of PSK.

Construction of the trellis and application of the Viterbi algorithm becomes more complicated for CPM signals other than MSK. The number of states depends on the modulation index $h$ and the extent of the pulse $g(t)$.

## 19.5. SCRAMBLING

Scrambling is a method of achieving d.c. balance and eliminating long sequences of zeros to ensure accurate timing recovery without redundant line coding. Scramblers use *maximum-length shift registers (MLSR)* on the input bit stream to "randomize" or "whiten" the statistics of the data, making it look more random.

Practical data transmission systems have no control over the bit sequences which the user is going to transmit. There are particular bit sequences, such as long strings of zeros or ones, which occur very often in practice and which can cause difficulties. At the theoretical level, these sequences strongly violate the assumption that the input sequence is random and i.i.d. On a more practical level, they can cause problems such as excessive radio frequency interference (RFI), crosstalk, and difficulty in timing recovery and adaptive equalization.

Any technique without redundancy such as scrambling must perform a one-to-one mapping between input data bit sequences and coded bit sequences. The objective is to map sequences that are problematic and fairly likely to occur (such as all zeros) into a coded sequence which looks more random and is less problematic. However, since the mapping is one-to-one, there must also be an input sequence that maps into a problematic sequence, such as all zeros! We

just hope that this input sequence is very improbable. Thus in general, redundant line coding is a safer method of achieving our desired objectives, but scrambling is attractive and often used on channels with extreme bandwidth constraints because it requires no redundancy.

**Example 19-29.** All CCITT-standardized voiceband data modems incorporate scrambling. This is attractive because of the desire to maximize spectral efficiency.

If a binary antipodal signal is wide-sense stationary, then the power spectral density after scrambling this signal is essentially flat down to d.c., although there is no discrete component at d.c. even if the original user bits have such a component (with some exceptions to be described later). This implies that an a.c. coupled medium is permissible, although the cutoff frequency has to be quite low to avoid appreciable baseline wander. Alternatively, some other line coding scheme, such as AMI, can be added to insert a rational zero at d.c. A combination of AMI and scrambling would be effective in eliminating low frequency components as well as insuring adequate timing energy. Often scrambling alone is combined with *quantized feedback* [waldhauer quantized feedback ] to compensate for the baseline wander. Quantized feedback is a form of decision-feedback equalization (Chapter 8) specifically designed to compensate for the baseline wander ISI using past decisions.

There are two forms of scrambling — *self-synchronizing* and *frame-synchronized*. Both types of scramblers use *maximal-length shift-register sequences*, which are periodic bit sequences with properties that make them appear to be random. These sequences are also called *pseudorandom sequences* because of their apparent randomness. A pseudorandom sequence generated by an $n$-bit shift-register is a binary sequence with period $r = 2^n - 1$.

Pseudorandom sequences are generated by a *feedback shift register* as pictured in Fig. 19-18. This device is governed by the relation

$$x_k = h_1 \cdot x_{k-1} \oplus \ldots \oplus h_n \cdot x_{k-n}\,, \tag{19.60}$$

where the summation is modulo-two, the output $x_k$ is binary assuming the values "0" and "1", and similarly the coefficients of the shift register are binary. The zero coefficients correspond to no feedback tap, whereas the one coefficients correspond to the direct connection of the shift register output to the modulo-two summation.

**Example 19-30.** A simple case is $n = 2$ and $h_1 = h_2 = 1$. For this case,

$$x_k = x_{k-1} \oplus x_{k-2}\,. \tag{19.61}$$

Adding $x_k$ to both sides of (19.60), and recalling that $x_k \oplus x_k = 0$, we get

$$x_k \oplus h_1 \cdot x_{k-1} \oplus \ \ldots \oplus h_n \cdot x_{k-n} = 0\,. \tag{19.62}$$

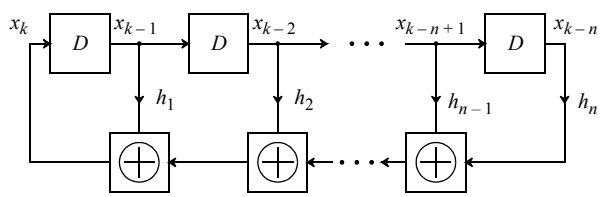



**Fig. 19-18.** A linear feedback shift register with binary input. The coefficients are binary, and the summation is modulo-two.

In other words,

$$x_k * h_k = 0 \,, \tag{19.63}$$

if we define $h_0 = 1$ and $h_m = 0$ for $m < 0$ and $m > n$ and of course we interpret the summation in the convolution in the modulo-two sense. The D-transform of (19.63) is

$$h(D)X(D) = 0 \,, \tag{19.64}$$

where

$$h(D) = 1 \oplus h_1 D \oplus \ldots \oplus h_n D^n \tag{19.65}$$

is the transfer function of the shift register. (19.65) is called a *modulo-two D-transform*, and is used extensively in Chapters 13 and 14. Given any binary sequence $b_k$ (deterministic or random), the modulo-two D-transform is

$$B(D) = \ldots \oplus b_{-1} D^{-1} \oplus b_0 \oplus b_1 D \oplus b_2 D^2 \oplus \ldots \tag{19.66}$$

where $\oplus$ denotes modulo-two addition. In other words, it is just like a Z-transform, except that the additions are modulo-two and the symbol $D$ is used instead of $z^{-1}$. Now any convolution of two sequences

$$C_k = g_k * B_k \tag{19.67}$$

can be written in the "D-domain" as

$$C(D) = G(D)B(D) \,. \tag{19.68}$$

The transfer function $h(D)$ for the generator is a polynomial of degree $n$ (we assume that $h_n = 1$) with binary coefficients, and is given the special name *generator polynomial*. There is a one-to-one correspondence between generator polynomials and feedback shift registers. Further mathematical properties of shift-register generators are discussed at some length in Appendix 19-A. Shift-register generators are also widely used as spreading sequences in spread spectrum (Section 6.4.3), because pseudo-random sequences have a constant-magnitude DFT (with the exception of the zero-frequency coefficient), making them suitable for generating broadband pulses with a narrow autocorrelation function.

### 19.5.1. Frame-Synchronized Scrambler

A frame-synchronized scrambler [22][23], also called a *cryptographic scrambler,* pictured in Fig. 19-19, performs a modulo-two summation of the user's bit stream $b_k$ with the output $x_k$ of a maximal-length feedback shift-register in the transmitter to generate the scrambled bit stream $c_k$,

$$c_k = b_k \oplus x_k \,. \tag{19.69}$$

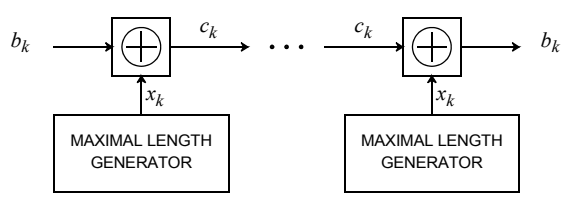



**Fig. 19-19.** A frame-synchronized scrambler where the maximal-length generator is as in Fig. 19-18.

The scrambled bit stream is transmitted to the receiver by whatever line coding method is chosen, where the stream is descrambled by another modulo-two summation with the output of another identical generator to recover the original user's bit stream. This recovery follows from the relation

$$c_k \oplus x_k = b_k \oplus x_k \oplus x_k = b_k \tag{19.70}$$

since $x_k \oplus x_k = 0$.

With binary antipodal signaling, if there were no scrambling then $b_k = 1$ would be transmitted as $a_k = +1$ and $b_k = 0$ would be transmitted as $a_k = -1$. Similarly, we can define a binary antipodal version of the generator sequence $x_k$, call it $s_k$, that assumes the values $\pm 1$. With scrambling, we substitute $c_k$ for $b_k$. We can easily see that this is equivalent to *multiplying* the binary antipodal version of the user's bits by the negative of the binary antipodal maximal-length sequence $s_k$; that is, we transmit $-s_k a_k$ in place of $a_k$.

The sequence $s_k$ is periodic, and hence consists of a sequence of equally spaced harmonics, where the spacing of harmonics is the sampling rate divided by the period $r$. In fact, it is shown in Appendix 19-A, Exercise 19-14, that the magnitude of all the harmonics, with the single exception of the d.c. harmonic, are equal ((19.90) and (19.94)). Thus, the scrambler is equivalent to modulating by a set of equal-amplitude harmonics equally-spaced across the band and summing the results. We would expect the result to be approximately white almost without regard to the spectrum of the original user bit stream. In quantifying this concept, we have to deal as always with the cyclostationary nature of a modulated signal.

**Exercise 19-8.** Assume the binary antipodal version $A_k$ of the user bit stream $B_k$ is a wide-sense stationary random process with autocorrelation function $R_a(l)$.

(a) Show that the scrambled sequence is cyclostationary.

(b) Average the autocorrelation over one period, and show that as $r \to \infty$ the power spectrum approaches $R_a(0)$. Hence, as $r$ gets large, the spectrum becomes white independent of the spectrum of the user's bit stream.

The correct operation of the frame-synchronized scrambler depends on the alignment in time of the two maximal-length sequences of period $r = 2^n - 1$ in the scrambler and descrambler. This is called *frame-synchronization*, and is accomplished by an additional frame synchronization mechanism.

If the user should provide the scrambler with the maximal-length sequence itself, the scrambled sequence will be all zeros! However, this eventuality should be very improbable, unless the user is deliberately attempting to sabotage his own transmission. More generally, the periodic structure of the generator output implies that the scrambled sequence will be periodic whenever the input stream is periodic.

**Exercise 19-9.** Show that when the input stream $b_k$ has period $s$ and the generator output has period $r = 2^n - 1$, then the scrambler output will be periodic with period equal to the least common multiple (LCM) of $s$ and $r$. (This does not preclude the period being a divisor of this LCM, as we will see below.) In particular, when $r$ is prime, which we can arrange, then this LCM period is $sr$, a multiple of the period of the maximal-length sequence.

We will generally maximize the period of the output, which is desirable, by choosing $r$ to be a prime number, such as $r = 2, 3, 7, 31, \ldots$. For this case, the smallest LCM will occur when $s = 1$, or in other words the input is all zeros or all ones.

A serious problem occurs when the input stream has period $r$ or some multiple of $r$, since the scrambled stream can then have a much shorter period than the LCM and can also have a large d.c. component resulting in severe baseline wander. If $r$ is large this pathological situation will arise with vanishingly low probability, and need not be of great concern.

## 19.5.2. Self-Synchronized Scrambler

We can avoid the necessity for frame synchronization of the scrambler by using the self-synchronized scrambler[22][23] of Fig. 19-20. For this case, we use a shift-register generator in the transmitter, except that we add the input stream directly to the input of the shift-register. The shift-register input $c_k$ is also the scrambled stream, and is applied to the input of an identical shift-register in the descrambler. Since both shift-registers, the one in the scrambler and the one in the descrambler, have the same inputs (in the absence of transmission errors), and the shift-register output is added modulo-two in the scrambler and descrambler, it follows that the input stream $b_k$ is recovered by the descrambler.

Mathematically, the scrambler is represented by the relation

$$c_k = b_k \oplus h_1 c_{k-1} \oplus \ldots \oplus h_n c_{k-n} , \tag{19.71}$$

and taking the D-transform of both sides we get

$$h(D)C(D) = B(D) , \tag{19.72}$$

where $h(D)$ is the same generator polynomial as in the maximal-length generator. Formally, we can write

$$C(D) = \frac{B(D)}{h(D)} , \tag{19.73}$$

and we can view the scrambler as dividing the polynomial corresponding to the input stream by the generator polynomial $h(D)$, whereas the descrambler multiplies the scrambled stream polynomial by $h(D)$, from (19.72).

**Example 19-31.** The CCITT V.22bis voiceband data modem uses a self-synchronizing scrambler with generating polynomial

$$h(D) = 1 \oplus D^{14} \oplus D^{17} . \tag{19.74}$$

The V.26ter modem uses two polynomials,

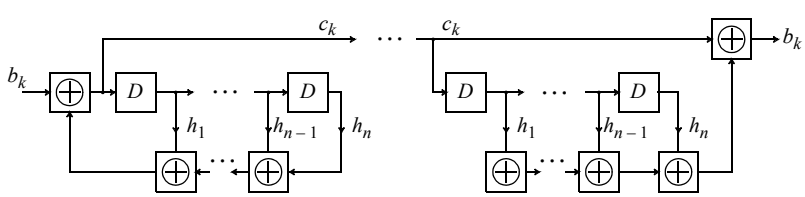



**Fig. 19-20.** A self-synchronized scrambler.

$$h(D) = 1 \oplus D^{18} \oplus D^{23} \tag{19.75}$$

in one direction of transmission and

$$h(D) = 1 \oplus D^{5} \oplus D^{23} \tag{19.76}$$

in the other direction. The reason for using two generators is that the V.26ter modem uses echo cancellation to separate the two directions of transmission (Chapter 20), and it turns out to be important to ensure that the scrambled streams in the two directions are uncorrelated.

Because of the linearity of the scrambler circuit, and in spite of the modulo-two arithmetic, we can view the processing as follows. The scrambler consists of an all-pole filter with transfer function $1/h(D)$; the descrambler consists of an all-zero filter with transfer function $h(D)$. The product of the two transfer functions is unity, recovering the original bit stream. The all-pole filter output can be decomposed into the superposition (modulo-two summation in this case) of two solutions — the zero-input solution (transient response) and the zero-state solution (steady-state response). The zero-input solution is precisely the maximal-length sequence used in the frame-synchronized scrambler, where this solution persists forever and does not die down as it might in a normal filter. In this view of the scrambler, the output is the all-pole filtered version of the input stream added to the maximal-length sequence. The latter gives us the "randomization" operation which we are attempting to achieve with the scrambler.

The self-synchronized scrambler works without any alignment of the scrambled sequence for reasons stated earlier. However, it does have one disadvantage — error propagation. When the input to the descrambler shift register is different from that of the scrambler shift register due to a transmission error, this causes additional errors. Specifically, there is one direct error and one secondary error for each non-zero tap in the shift register. The error multiplication is therefore by a factor equal to one plus the number of non-zero taps.

The self-synchronizing scrambler also has more problems with periodic input streams than the frame-synchronized scrambler [22]. Specifically, when the input has period $p$, the output will have one of the following periods depending on the initial state. For one particular state (with probability $2^{-n}$), the period will be $p$, which could be very unfortunate when $p$ is very small. For the remaining states (with probability $1 - 2^{-n}$), the output has period equal to the LCM of $p$ and $r$, the same as the frame-synchronized scrambler.

**Example 19-32.** Using the second-order generator polynomial of Example 19-30, the scrambler is given by

$$c_k = b_k \oplus c_{k-1} \oplus c_{k-\delta 2} \ . \tag{19.77}$$

Assume that we start a periodic input just at the point that the state happens to be $(c_{k-1}, c_{k-2}) = (0, 0)$. Then we can see that the the response to the all-zero input from that point is all-zero (the same period, $p = 1$, as the input). For any other initial state the output has period $1 \cdot 3 = 3$. As another example, assume the input is alternating zero-one (period two). Then for an initial state (0,1) the output is also alternating zero-one (period two), whereas for any other state the output has period $2 \cdot 3 = 6$.

The probability of a short periodic output can be minimized by choosing $n$ large.

## 19.6. FURTHER READING

Line coding is a practical subject that is not widely covered in textbooks. One excellent reference is the tutorial article by Duc and Smith [4]. There is some coverage in a recent digital communications textbook [24]. Recent results in combining line coding with trellis coding are reviewed in [2], and recent results in line codes for magnetic recording are discussed in [25].

Partial response is covered in a tutorial paper [26] and book chapter [27]. The performance of the Viterbi algorithm for partial response systems was determined by Kobayashi [28] and Forney [29].

A general treatment of full-response continuous-phase modulation is given by Aulin and Sundberg [30], and of partial response CPM by Aulin, Rydbeck, and Sundberg [20]. A general discussion is also given by Simon [31]. For a general approach to modeling the phase evolution as a Markov chain, see [32]. The same issue of the IEEE Transactions on Communications (March 1981) has several useful papers on CPM in a special section. For tutorials on MSK, we recommend Pasupathy [33] and Haykin [34].

Finite fields and maximal-length shift register sequences are covered in detail in [35].

# Appendix 19-A. Maximal-Length Feedback Shift Registers

In this appendix we will consider the properties of a periodic sequence generated by the shift register circuit of Fig. 19-18 with generator polynomial $h(D)$. While a full treatment of this problem requires some sophisticated mathematics, we can understand most of the properties of this generator using only elementary concepts.

Mathematically, the binary coefficients of the generator polynomial together with the rules of multiplication and modulo-two addition constitute an *algebraic field*, similar to the real or complex numbers. In recognition that this field has only two elements, it is also called a *finite field* or *Galois field with two elements GF(2)*. In general there exists a single finite field with a number of elements equal to any prime integer to any power. We will limit ourselves here to finite fields with two elements, which is just the modulo-two arithmetic considered in this chapter. The more general case is discussed in Appendix 12-A. GF(2) has two elements "0" and "1", and modulo-two arithmetic is used.

**Example 19-33.** As an illustration of polynomial arithmetic over GF(2), multiplying the polynomials $(1 \oplus D)$ and $(1 \oplus D \oplus D^2)$,

$$(1 \oplus D)(1 \oplus D \oplus D^2) = 1 \oplus D \oplus D \oplus D^2 \oplus D^2 \oplus D^3 = 1 \oplus D^3 \,. \tag{19.78}$$

We have used the fact that, for example,

$$D \oplus D = (1 \oplus 1)D = 0 \cdot D = 0 \;. \tag{19.79}$$

We know that $n$-th order polynomials with real-valued coefficients always have $n$ roots, but only if we allow those roots to be complex-valued. In general a polynomial with real coefficients cannot always be factored into a product of lower-order polynomials with real coefficients (for example $X^2 + 1$). Similarly, a GF(2) polynomial cannot always be factored into two or more polynomials with GF(2) coefficients.

**Example 19-34.** Continuing Example 19-33, the polynomial $(1 \oplus D \oplus D^2)$ cannot be factored into the product of two first order polynomials over GF(2). In fact, the only first-order polynomials over GF(2) are $D$ and $(1 \oplus D)$, and the reader can readily verify that they cannot be factors of $(1 \oplus D \oplus D^2)$.

A polynomial that has no factors other than itself and 1 is called an *irreducible polynomial over GF(2)*. In the sequel we will assume that the generator polynomial $h(D)$ is irreducible.

Returning to the feedback shift-register, the state $(x_{k-1}, \ldots, x_{k-n})$ can assume at most $2^n$ distinct values. From this fact, and other properties of the register, we can discern the following properties:

- If the state of the shift-register is all-zero (00 … 0) at any time, then it must always be all-zero. Thus, we must ensure that this state is never visited unless we are satisfied with a complicated circuit that just generates all-zeros at the output.
- If the state ever stays the same from one time increment to the next, then it will forever be the same. Thus, if the output is to be interesting (anything but all-zeros or all-ones), then we must ensure that the state always changes upon every time increment.
- The sequence of states must be periodic. Since there are only $2^n$ distinct states, the sequence of states must always return to an initial state, after which the sequence of states repeats. Since the output $x_k$ is a function of the state, it must also be periodic.
- Combining the first three, the maximum period of the states and outputs must be $(2^n - 1)$ time increments. This maximum period would correspond to a periodic sequence of states which change at every time increment and which cycle through every state except the all-zero state.

A feedback shift-register is called *maximal-length* if the period of the output is $r = 2^n - 1$.

**Example 19-35.** The generator polynomial for the shift-register of Example 19-30 is $h(D) = 1 \oplus D \oplus D^2$. From Example 19-34 this generator polynomial is irreducible. We can verify that the shift register is maximal-length, that is has period $2^2 - 1 = 3$. Starting with state (0,1), the following table specifies the state and output vs. time for four cycles:

| $x_k$ | $x_{k-1}$ | 1 |
|---|---|---|
| 1 | 0 | 1 |
| 1 | 1 | 0 |
| 0 | 1 | 1 |
| 1 | 0 | 1 |

Note that the state has returned to its initial value at the fourth time increment, and therefore the shift-register will continue with the same sequence of states. Also note that if we initialized the state with any of the other two values, the same sequence of states would result, but we would just start at a different point in the sequence.

We could easily envision that the period of a shift-register sequence could be less than $2^n$ 1 in length.

**Exercise 19-10.** Show that the shift-register sequence corresponding to polynomial $h(D) = 1 \oplus D^2$ has period one or two depending on the initial state.

We would like to have some criterion to establish when a generator polynomial corresponds to a maximal-length shift-register sequence. When an irreducible polynomial $h(D)$ of degree $n$ does not divide any polynomial $(1 \oplus D^m)$ for $m < 2^n - 1$, it is said to be *primitive*. A shift-register sequence is maximal-length if and only if the generator polynomial is primitive [36].

**Example 19-36.** We can verify that the generator polynomial of Example 19-30, $h(D) = 1 \oplus D \oplus D^2$, is primitive. This is because it obviously does not divide $(1 \oplus D^2)$, while it does divide $(1 \oplus D^3)$ since

$$(1 \oplus D \oplus D^2)(1 \oplus D) = 1 \oplus D^3 \tag{19.80}$$

from Example 19-33.

Fortunately, there exist primitive polynomials of all orders. The polynomials with *minimum weight*, that is with the minimum number of shift-register taps, of all orders up to $n = 34$ are listed in Table 19-2.

**Example 19-37.** A maximal-length shift-register of order 12 can be found from Table 19-2. The octal entry is "10123," which corresponds to binary "1000001010011" and hence polynomial

$$h(D) = 1 \oplus D \oplus D^4 \oplus D^6 \oplus D^{12} . \tag{19.81}$$

Hence the shift-register is characterized by difference equation

$$x_k = x_{k-1} \oplus x_{k-4} \oplus x_{k-6} \oplus x_{k-12} . \tag{19.82}$$

**Table 19-2.** Minimal weight primitive polynomials of orders two through 34 [36]. Each entry in the table is an octal number, which when converted to binary specifies the coefficients of the polynomial $h(D)$. The most significant (left-most) bit is $h_n = 1$ and the least significant (right-most) bit is $h_0 = 1$.

| Order | Polynomial | Order | Polynomial |
|---|---|---|---|
| 2 | 7 | 19 | 2000047 |
| 3 | 13 | 20 | 4000011 |
| 4 | 23 | 21 | 10000005 |
| 5 | 45 | 22 | 20000003 |
| 6 | 103 | 23 | 40000041 |
| 7 | 211 | 24 | 100000207 |
| 8 | 435 | 25 | 200000011 |
| 9 | 1021 | 26 | 400000107 |
| 10 | 2011 | 27 | 1000000047 |
| 11 | 4005 | 28 | 2000000011 |
| 12 | 10123 | 29 | 4000000005 |
| 13 | 20033 | 30 | 10040000007 |
| 14 | 42103 | 31 | 20000000011 |
| 15 | 100003 | 32 | 40020000007 |
| 16 | 210013 | 33 | 100000020001 |
| 17 | 400011 | 34 | 201000000007 |
| 18 | 1000201 | | |

An interesting property of maximal-length sequences is that if we look at $n$-bit segments of the sequence, we will see all possible $n$-bit words, with the exception of the all-zero word. This follows from the fact that the state of the shift-register passes through all possibilities except all-zeros, and the state is equal to the past $n$ bits of the output. The maximal-length sequence therefore satisfies a minimal condition for "randomness," since we would expect to see all combinations of bits (except the all-zero) in such a sequence.

The output of a maximal-length shift register is often called a *pseudorandom sequence*. This is because, even though the sequence is deterministic and periodic, it displays many of the properties of a random sequence (analogous for example to a numerical algorithm for random number generation). We can see these properties reflected in the *relative frequency* and in the *autocorrelation function*.

The relative frequency of observing particular sequences of $i$ bits in a maximal-length sequence is close to the probability of observing the $i$ bits in an i.i.d. random sequence as long as $i \leq n$, since all possible sequences of $n$ bits occur once in one period of $2^n - 1$ bits, with the exception of the all-zero sequence.

**Exercise 19-11.** Show that the relative frequency of any particular sequence of $i \leq n$ bits in the maximal-length sequence is

$$\frac{2^{n-i}}{2^n - 1} \approx 2^{-i} \tag{19.83}$$

for the case where the $i$ bits do not constitute the all-zero sequence, and for the all-zero sequence of $i$ bits

$$\frac{2^{n-i} - 1}{2^n - 1} \approx 2^{-i}. \tag{19.84}$$

The approximations apply to large $n$, and hence for this case the sequence looks random on a relative frequency basis as long as we don't observe blocks of bits greater than $n$.

The autocorrelation function can be determined using the *cycle-and-add property* of the maximal-length sequence [37]. This property says that if we modulo-two add the maximal-length sequence to itself, where one of the sequences has been shifted in time, we get another version of the same sequence shifted in time,

$$x_k \oplus x_{k+l} = x_{k+j} \tag{19.85}$$

for $l \in \{0, 1, \ldots, r-1\}$, where $j$ depends on $l$. Of course, when $l = 0$ the sum is the all-zero sequence (this is a degenerate case of a maximal-length sequence). The cycle-and-add property follows from the fact that if $h(D)X(D) = 0$, then obviously $(1 \oplus D^l)h(D)X(D) = 0$, and therefore $(1 \oplus D^l)X(D)$ must also have generator polynomial $h(D)$. Many interesting properties can be derived from (19.85).

**Example 19-38.** Since $x_k \oplus x_{k+l} = 0$ if and only if $x_k = x_{k+l}$, it follows that $x_k = x_{k+l}$ for precisely $(r-1)/2$ values of $k$ within one period $k \in \{0, 1, \ldots, r-1\}$, and $x_k \neq x_{k+l}$ for precisely $(r+1)/2$ values of $k$. Again, $r = 2^n - 1$ is the length of the sequence.

In terms of the autocorrelation, we are usually interested in the autocorrelation of a binary antipodal sequence $s_k$ obtained by mapping $x_k = 0$ into $s_k = -1$ and $x_k = 1$ into $s_k = +1$. We will call this new sequence the *binary antipodal maximal-length sequence*. The autocorrelation function of this sequence is defined as

$$R_s(l) = \frac{1}{r}\sum_{k=0}^{r-1} s_k s_{k+l}. \tag{19.86}$$

This is a *time-average* autocorrelation function averaged over one period of the sequence. Of course it is a periodic function of $l$, and hence we need only be concerned with the value for $l \in \{0, 1, \ldots, r-1\}$. Similarly, we can define a time-average mean value of the sequence as

$$\mu_s = \frac{1}{r}\sum_{k=0}^{r-1} s_k \ . \tag{19.87}$$

**Exercise 19-12.** Using the relative frequency property, show that

$$\mu_s = \frac{1}{r} \tag{19.88}$$

which approaches zero as $n$ (and hence $r$) gets large.

**Exercise 19-13.** Use the cycle-and-add property to show that the autocorrelation function is given by

$$R_s(l) = \begin{cases} 1, & l \neq 0 \\ -1/r, & l \in \{1, 2, \ldots, r-1\} \end{cases} \ . \tag{19.89}$$

Hence, when $r$ is large, the time-average autocorrelation function approaches zero except at multiples of the period. Except for the periodicity, this approaches the autocorrelation of a white sequence, and hence is another indication of the pseudo-random property.

Using this time-average autocorrelation, we can infer another important property of the binary antipodal maximal-length sequence; namely, its harmonic structure. Since this sequence is periodic, we can expand it using a DFT,

$$s_k = \frac{1}{r}\sum_{m=0}^{r-1} S_m e^{j2\pi mk/r}, \tag{19.90}$$

where

$$S_m = \sum_{k=0}^{r-1} s_k e^{-j2\pi mk/r}, \qquad m \in \{0, 1, \ldots, r-1\} \ . \tag{19.91}$$

We can easily relate the harmonics of the sequence to the autocorrelation function.

**Exercise 19-14.**

(a) Show that

$$\sum_{k=0}^{r-1} s_{k+l}\, e^{-j2\pi mk/r} = e^{j2\pi mk/r} \sum_{k=0}^{r-1} s_k\, e^{-j2\pi mk/r}. \tag{19.92}$$

(b) Show that

$$\frac{1}{r}|S_m|^2 = \sum_{l=0}^{r-1} R_s(l)\, e^{-j2\pi mk/r}. \tag{19.93}$$

(c) Evaluate this DFT to show that

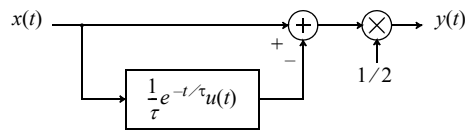
$$|S_m|^2 = \begin{cases} 1, & m = 0 \\ r+1, & m \in \{1, 2, \ldots, r-1\} \end{cases}. \tag{19.94}$$

Hence, the harmonics of the sequence are all equal to one another in magnitude, except for the d.c. component, which is relatively small. This resembles the power spectrum of a white sequence, and this property makes these sequences desirable as spreading sequences in spread spectrum.

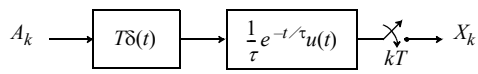
# Problems

**Problem 19-1.** Consider the a.c. coupled circuit in Fig. 19-1a.

(a) Show that an equivalent representation of the circuit in terms of a linear time-invariant system is shown below:




(b) Assume an input PAM signal with delta-function pulses with area $T$ (since real PAM pulses have width of order $T$), yielding the equivalent system below (assuming symbol-rate sampling in the receiver):



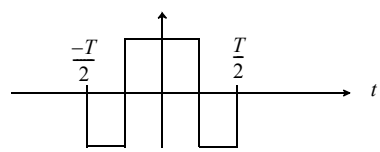

Calculate the output ISI, thereby establishing the equivalency of Fig. 19-1b.

**Problem 19-2.**

(a) What is the largest possible ISI in Fig. 19-1b due to baseline wander, where no provision has been made to counter baseline wander in the line code?

(b) Evaluate this maximum ISI for a.c. coupling cutoff frequency equal to 1% of the symbol rate.

**Problem 19-3.** Repeat Exercise 19-1 for the *Wal2 pulse shape*, a binary antipodal line code with the pulse shape shown below:




Evaluate the intersymbol interference for the same β as in Example 19-2.

**Problem 19-4.** Assume that a baseband PAM system uses a transmitted biphase pulse, and through a combination of transmit filtering and receive equalization we equalize to a pulse shape that satisfies the Nyquist criterion.

(a) Show that such an equalized pulse must have an excess bandwidth greater than 100%. Thus, the minimum bandwidth for an equalized biphase pulse shape is twice that required for transmitted pulses that are allowed to have a d.c. content.

(b) Explicitly write the Nyquist criterion for an equalized biphase pulse with excess bandwidth greater than 100% and less than 150%.

(c) What is the equalized biphase pulse shape with the minimum bandwidth that satisfies the Nyquist criterion?

(d) Does the result of a. apply to all transmitted pulses which have zero area?

**Problem 19-5.** Keeping the average transmitted power the same, compare the noise immunity for a twinned binary code and a binary antipodal code. Assume the input information bits are independent, and that a "one" has probability $p$. Assume the same pulse shape in both cases, assume all time-translates of the basic pulse are orthogonal, and be sure to take into account the transmitted signal power and the fact that in the twinned binary code the transmitted levels are not equally likely.

(a) Show that for $p = 1/2$, the noise immunity of the binary antipodal code is 3 dB better.

(b) Compare the noise immunity for the same average transmitted power for arbitrary $p$. For what range of $p$ does the antipodal code have better noise immunity?

**Problem 19-6.** Interpret the twinned binary code as a binary antipodal transmitted data symbol with a transmitted pulse shape that is two symbol intervals wide.

**Problem 19-7.** For a twinned binary code, show how a DFE could be used in the receiver in place of the receiver structure of Fig. 19-3. Compare the noise immunity of this approach to that of the conventional interpretation ( Problem 19-5).

**Problem 19-8.** Show how the Viterbi algorithm can be used to decode the twinned binary code, and determine the advantage in $\Xi_{VA}$, the argument of $Q(\cdot)$, that can be obtained.

**Problem 19-9.** Show that the Viterbi algorithm can be used to advantage to decode an AMI-coded signal. You will have to make an assumption about the noise statistics at the AMI slicer input. What is the improvement in $\Xi_{VA}$, the argument of $Q(\cdot)$?

**Problem 19-10.** In an AMI decoder, a *bipolar violation* is the term for a violation of the known constraints on sequences of ternary levels. The term arises because AMI line coding is sometimes called bipolar line coding.

(a) List all possible bipolar violations.

(b) Describe how bipolar violations can be used for in-service monitoring.

(c) Make a table that relates the number of bit errors to the number of bipolar violations for all single errors in slicing of the ternary signal. From this table estimate the relationship between the rate of bit errors and bipolar violations.

**Problem 19-11.** Adopt the following notation for a pseudoternary line code. A transmitted "0" is just that, a transmitted "B" is a non-zero symbol which obeys the AMI constraint (it is opposite in polarity to the last transmitted non-zero symbol), and a transmitted "V" is a non-zero symbol that violates this constraint (has the same polarity as the last transmitted non-zero symbol).

(a) Describe an AMI line code in these terms.

(b) Give an example of a signal containing "V"'s that is still d.c. balanced. What price do we pay for introducing "V"'s?

**Problem 19-12.** Using the notation of Problem 19-11, the B6ZS (*bipolar six-zero substitution*) code substitutes at the output of an AMI coder, for each block of six "zeros," the code word "B0VB0V." Specifically then, the transmitted block of six symbols would be "+ 0 + – 0 –" for an RDS of zero at the start of the block (last transmitted non-zero symbol was "–"), and "– 0 – + 0 +" otherwise.

(a) What is the advantage of this?

(b) Describe the decoder.

(c) What is the RDS of this code?

**Problem 19-13.** An alternative to the B6ZS code of Problem 19-12 is the HDBk (*High-Density Bipolar*) code, which achieves a lower DSV. The code word "B00…0V" or "00 …0V" is substituted for a block of $k + 1$ "zeros," where each of these code words is $k + 1$ symbols long. The appropriate code word is chosen so as to make the number of "B"'s between consecutive "V"'s odd. Note that the two code words allow us to put in a "B" or not, and hence we can control whether the number of "B"'s is even or odd by the choice of the code word. For example, in HDB3, if the number of "B"'s since the last "V" is even at the beginning of the block of four "zeros," then we transmit block "B00V." Otherwise, we transmit "000V." Show that the RDS of HDBk is limited to the range $-1 \leq RDS \leq +1$, and hence the DSV is two. The penalty relative to AMI in baseline wander is therefore small. (*Hint:* Consider the disparity of a block of symbols starting at one "V" and extending up to the next "V.")

**Problem 19-14.**

(a) Make a reasonable definition of the code B4ZS (see Problem 19-12).

(b) What is the RDS and DSV of this code?

(c) What advantages or disadvantages might this code have over B6ZS?

**Problem 19-15.**

(a) Show that there is no B3ZS code similar to B6ZS defined in Problem 19-12.

(b) Show that by introducing two modes into the code, a B3ZS code can be defined.

**Problem 19-16.** For the 4B3T line code described in this chapter, describe how in-service monitoring of error rate could be performed at the decoder.

**Problem 19-17.** Define the *one's density* of a pseudoternary code as follows: for each $n \geq 1$ it is the smallest value the quantity *movern* can assume, where $n$ is the number of symbols in a block and $m$ is the number of non-zero symbols in this block. Plot this quantity for AMI, B6ZS, HDB3, and 4B3T.

**Problem 19-18.** Design a bimode binary block code which maps three information bits into four transmitted binary data symbols (75% efficiency) and maintains a DSV less than or equal to four.

**Problem 19-19.** Modify the results of Example 19-15 to yield a first-order spectral null at $z = -1$, half the sampling rate.

**Problem 19-20.** Show by example that if the ternary slicer in Fig. 19-12c makes an error, this error can propagate. Under what conditions is this propagation the biggest problem?

**Problem 19-21.** *Class II partial response.* Given a partial response system with polynomial $F(D) = (1 + D)^2$.

(a) Draw a typical transmitted pulse shape and describe qualitatively what is accomplished by using this shape.

(b) Give a truth table for the precoder and give a Boolean logic expression for the precoder design.

(c) How many levels does the slicer have? Specify the decoder.

**Problem 19-22.** Let the input bits be independent and identically distributed, with the probability of a "one" equal to $p$. The power spectrum for the dicode PR output symbols was determined (equivalently for AMI) in Section 19.2. Determine this output spectrum for duobinary and modified duobinary.

**Problem 19-23.** Given a discrete-time channel given by

$$Y_k = X_k + \rho X_{k-1} + N_k \tag{19.95}$$

where $0 < \rho < 1$ and the additive noise is white and zero-mean with variance $\sigma^2$, derive the error probability or bounds on the error probability expressed in terms of both the peak and average transmitted power at the transmitter output for the following cases:

(a) We use *receiver LE-ZF equalization* and binary antipodal signaling.

(b) Same as part (a), except we use *transmitter equalization*.

(c) We use *duobinary PR* precoding in the transmitter, and in the receiver we equalize to a $(1 + D)$ response prior to the three-level slicer.

(d) Same as part (c), except we do the equalization in the transmitter.

(e) We use binary antipodal signaling together with ML sequence estimation in the receiver.

**Problem 19-24.** Specify a general scheme to use the redundancy inherent in a PR encoded signal to do performance monitoring (unreliable error detection) at the slicer output. Specify this scheme specifically for dicode and duobinary PR with a three-level slicer, and relate to earlier results in Section 19.1.

**Problem 19-25.** Generalize PR from the binary case considered in the chapter to an input PAM signal with $M$ equally-spaced levels.

**Problem 19-26.** Consider the MSK signal in (19.57).

(a) Using Fig. 19-17 as a starting point, draw a trellis with a finite number of states that describes the phase evolution of the MSK signal.

(b) Show that the minimum-distance error event is the error event of length one. Find its distance.

(c) Compare the optimal sequence detector performance to that of the receiver in Fig. 6-25.

**Problem 19-27.** For a frame-synchronized scrambler, find a pathological input bit stream with period equal to $r$ that results in a scrambled sequence with period two.

**Problem 19-28.** Use Table 19-2 to design a maximal-length shift-register of order $n = 3$. Calculate the sequence of states and outputs to verify that the period is $2^3 - 1 = 7$.

**Problem 19-29.** Repeat Problem 19-28 for $n = 4$.

# References


1. P. A. Franaszek, “Sequence-State Coding for Digital Transmission,” *Bell System Technical Journal*, Vol. 47, p. 143 (Jan. 1968).
2. A. R. Calderbank and J. E. Mazo, “Spectral Nulls and Coding with Large Alphabets,” IEEE Communications Magazine, (Dec. 1991).
3. S. Yoshida and S. Yajima, “On the Relationship Between Encoding Automaton and the Power Spectrum of its Output Sequence,” *Trans. IECE* (Japan), Vol. E59, p. 97 (1976).
4. N. Q. Duc and B. M. Smith, “Line Coding for Digital Data Transmission,” *Australian Telecommunications Research* (A. T. R), Vol. 11 (2), (1977).
5. A. Croisier, “Introduction to Pseudoternary Transmission Codes,” *IBM J. Research and Development*, Vol. 14, p. 354 (July 1970).
6. A. Brosio, U. DeJulio, V. Lazzari, R. Ravaglia, and A. Tofanelli, “A Comparison of Digital Subscriber Line Transmission Systems Employing Different Line Codes,” *IEEE Trans. on Communications*, Vol. COM-29 (11), p. 1581 (Nov. 1981).
7. L. A. Meacham, “Twinned Binary Transmission,” U.S. Patent 2,759,047.
8. M. R. Aaron, “PCM Transmission in the Exchange Plant,” *Bell System Technical Journal*, Vol. 41, pp. 99-141 (Jan. 1962).
9. H. Sailer, H. Schenk, and E. Schmid, “A VLSI Transceiver for the ISDN Customer Access,” *Proc. IEEE Int. Conf. Communications*, (June 1985).
10. R. F. Lyon, “Two-Level Block Encoding for Digital Transmission,” *IEEE Trans. on Communications*, Vol. COM-21 (12), p. 1438 (Dec. 1973).
11. J. N. Franklin and J. R. Pierce, “Spectra and Effi- ciency of Binary Codes without DC,” *IEEE Trans. on Communications*, Vol. COM-20 (6), p. 1182 (Dec. 1972).
12. J. K. Wolf, “Modulation and Coding for the Magnetic Recording Channel,” *Proceedings NATO Advanced Study Institute*, (July 1986).
13. H. Kobayashi, “A Survey of Coding Schemes for Transmission or Recording of Digital Data,” *IEEE Trans. on Communications*, Vol. COM-19, p. 1087 (Dec. 1971).
14. G. D. Forney, Jr and A. R. Calderbank, “Coset Codes for Partial Response Channels; Or Coset Codes with Spectral Nulls,” *IEEE Trans. on Information Theory*, Vol. IT-35, p. 925 (1989).
15. A. R. Calderbank and J. E. Mazo, “Baseband Line Codes Via Spectral Factorization,” *IEEE Trans. on Selected Areas of Communications*, p. 914 (Aug. 1989).
16. D. G. Messerschmitt, “Generalized Partial Response for Equalized Channels with Rational Spectra,” *IEEE Trans. on Communications*, Vol. COM-23 (11), p. 1251 (Nov. 1975).
17. A. Lender, “The Duobinary Technique for High-Speed Data Transmission,” *IEEE Trans. on Commun. Electronics*, Vol. 7 (March 1963).
18. E. Kretzmer, “Generalization of a Technique for Binary Data Communication,” *IEEE Trans. on Communication Tech.*, Vol. COM-14, (Feb. 1966).
19. D. G. Messerschmitt, “Design of a Finite Impulse Response for the Viterbi Algorithm and Decision Feedback Equalizer,” *Proc. IEEE Int. Conf. on Communications*, (June 1974).

20. T. Aulin, N. Rydbeck, and C.-E. W. Sundberg, "Continuous Phase Modulation - Part II: Partial Response Signaling," *IEEE Trans. on Communications*, Vol. COM-29 (3), (March 1981).

21. F. D. Waldhauer, "Quantized Feedback in an Experimental 280-Mb/s Digital Repeater for Coaxial Transmission," *IEEE Trans. on Communications*, Vol. COM-22 (1), p. 1 (Jan. 1974).

22. J. E. Savage, "Some Simple Self-Synchronizing Digital Data Scramblers," *Bell System Technical Journal*, Vol. 46 (2), p. 449 (Feb. 1967).

23. D. G. Leeper, "A Universal Digital Data Scrambler," *Bell System Technical Journal*, Vol. 52 (10), p. 1851 (Dec. 1973).

24. S. Benedetto, E. Biglieri, and V. Castellani, Digital Transmission Theory, Prentice-Hall, Inc., Englewood Cliffs, NJ (1987).

25. P. H. Siegel and J. K. Wolf, "Modulation and Coding for Information Storage," IEEE Communications Magazine, (Dec. 1991).

26. P. Kabal and S. Pasupathy, "Partial-Response Signaling," *IEEE Trans. on Communications*, Vol. COM-23 (9), (Sep. 1975).

27. S. Pasupathy, "Correlative Coding: Baseband and Modulation Applications," pp. 429 in *Advanced Digital Communications Systems and Signal Processing Techniques*, ed. K. Feher,Prentice-Hall, Englewood Cliffs, N.J. (1987).

28. H. Kobayashi, "Correlative Level Coding and Maximum-Likelihood Decoding," IEEE Trans. Inform. Theory, Vol. IT-17, pp. 586-594 (Sept. 1971).

29. G. D. Forney, Jr., "Maximum-Likelihood Sequence Estimation of Digital Sequences in the Presence of Intersymbol Interference," *IEEE Trans. on Information Theory*, Vol. IT-18, pp. 363-378 (May 1972).

30. T. Aulin and C.-E. W. Sundberg, "Continuous Phase Modulation - Part I: Full Response Signaling," *IEEE Trans. on Communications*, Vol. COM-29, (3), (March 1981).

31. M. K. Simon, "A Generalization of Minimum-Shift-Keying (MSK)-Type Signaling Based Upon Input Data Symbol Pulse Shaping," *IEEE Trans. on Communications*, Vol. COM-24 (8), (Aug. 1976).

32. J. B. Anderson, C.-E. W. Sundberg, T. Aulin, and N. Rydbeck, "Power-Bandwidth Performance of Smoothed Phase Modulation Codes," *IEEE Trans. on Communications*, Vol. COM-29 (3), (March 1981).

33. S. Pasupathy, "Minimum Shift Keying: A Spectrally Efficient Modulation," *IEEE Communications Magazine*, Vol. 17 (4), (July 1979).

34. S. Haykin, *Communication Systems*, 2nd Edition, John Wiley & Sons, Inc. (1983).

35. R. J. McEliece, *Finite Fields for Computer Scientists and Engineers*, Kluwer Academic Publishers, Norwell, Mass. (1987).

36. W. Peterson and E. Weldon, *Error-Correcting Codes*, 2nd Ed., M.I.T. Press, Cambridge, Mass (1972).

37. S. W. Golomb, *Shift Register Sequences*, Holden-Day, San Francisco (1967).