Chapter 1

Channels and Constraints

The purpose of this chapter is to motivate, via several engineering examples, the subject of coding for constrained systems. Beyond this chapter the text will take on a more mathematical flavor.

1.1 Magnetic Recording

In this section we give a very brief introduction to magnetic recording. For more background on this subject, the reader is referred to Section 1.6 and the book [WT99].

The essential components of a magnetic recording system are a recording head, a read-head and a recording medium, such as a rotating magnetic disk. The disk is divided into concentric tracks. To write a string of data along a track, the recording head is positioned above the track, and current sent through the head magnetizes the track in one of two directions called magnetic polarities. This process is illustrated in Figure 1.1. A clock runs at a constant bit period of T seconds, and at each clock tick (i.e., at each multiple of T), the recording head has the opportunity to change the polarity on the disk: a 1 is recorded by changing the direction of the current, while a 0 is recorded by *not* changing the direction of the current (i.e., doing nothing). In this way, a 1 is represented as a transition in magnetic polarity along a data track, while a 0 is represented as an absence of such a transition. So, we can think of any binary sequence as a sequence of transitions/no-transitions; when we want to emphasize that we are thinking of a binary sequence in this way, we will call it a *transition sequence*.

In reading mode, the read-head, positioned above a track on the rotating disk, responds to a magnetic transition by an induced voltage; naturally, the absence of a transition produces no voltage response in the read-head. A bit cell is a window of length T centered on a clock tick. When a sufficiently high (positive or negative) voltage peak is detected within a bit

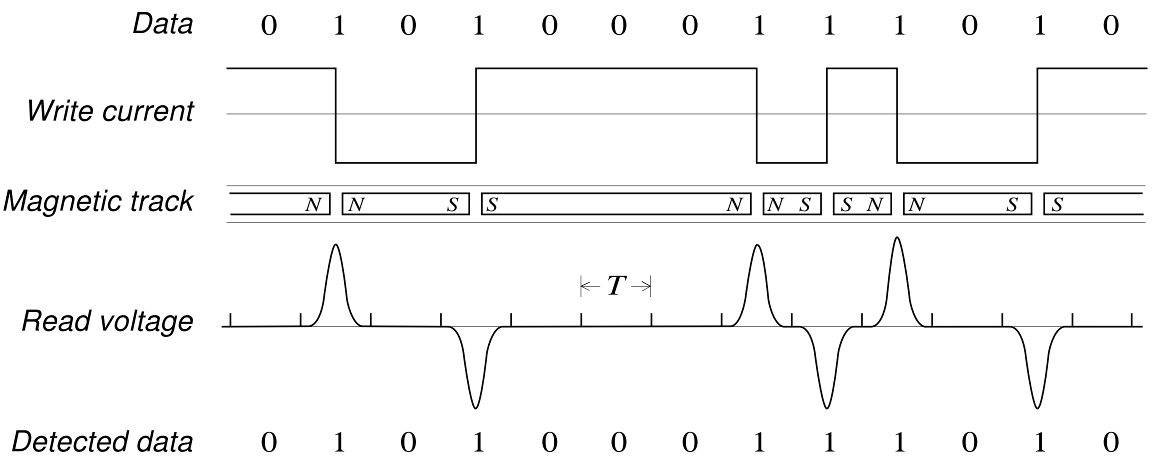

Figure 1.1: Digital magnetic recording.

cell, a 1 is declared; otherwise, a 0 is declared. This scheme is known as *peak detection*. Note that as shown in Figure 1.1, the readback voltages corresponding to successive transitions are of opposite signs; however, in most implementations the peak detector ignores the signs. We also remark that due to the presence of noise, the readback voltage signal in reality is not as smooth as Figure 1.1 might suggest.

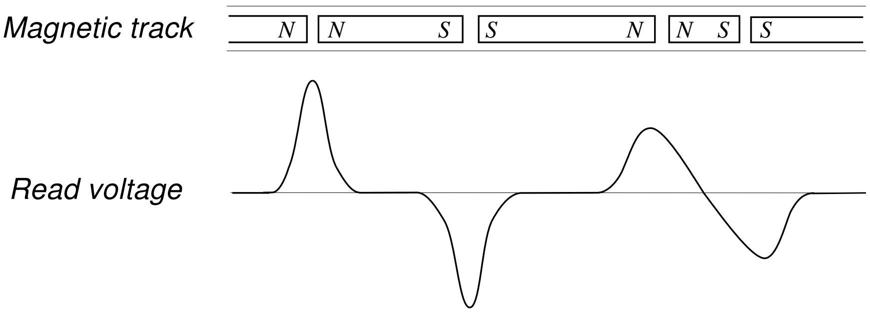

Figure 1.2: Inter-symbol interference.

Now, if successive 1's are too close together, then the voltage responses corresponding to the magnetic transitions may interfere with one another; this is known as *inter-symbol interference*. From Figure 1.2, which shows a reconstruction of the received signal with interference, it is evident that such interference may degrade the signal so that either one or both transitions are missed. In such a case, a recorded 1 will be mis-read as a 0; perhaps, even worse, the bit cell corresponding to a transition may be incorrectly identified so that a recorded 01 will be mis-read as 10; such an error is called a *peak shift* or *bit shift*. The likelihood of these kinds of errors can be reduced if 1's are separated sufficiently far apart, or equivalently if any two 1's are separated by a sufficiently long run of 0's.

Determination of the correct runlength of 0's between two successive 1's depends critically on accurate clocking. But accurate clocking can well be compromised by various imperfections in the recording system, such as variations in speed of the rotating disk. Thus, clocking needs to be adjusted periodically via a timing control scheme, which adjusts the clock in response to identification of peaks in the received signal. For instance, if a peak is supposed to occur at exactly the mid-point of the bit cell, but instead occurs near the end of the bit cell, then the next clock tick should be delayed. But during a long run of 0's, the ideal received signal will not contain any peaks. Hence, for timing control, it is desirable to avoid long runs of 0's.

This discussion suggests that it may be helpful to impose constraints on the binary sequences that are actually recorded.

1.2 Classical runlength constraints

The *(d,k)-runlength-limited (RLL) constraint* is described as follows.

Let d and k be integers such that $0 \leq d \leq k$. We say that a finite length binary sequence $\mathbf{w}$ satisfies the (d,k)-RLL constraint if the following two conditions hold:

- the runs of 0's have length at most k (the *k-constraint*), and —
- the runs of 0's between successive 1's have length at least d (the *d-constraint*); the first and last runs of 0's are allowed to have lengths smaller than d.

According to the discussion in Section 1.1, for binary transition sequences the d-constraint reduces the effect of inter-symbol interference and the k-constraint aids in timing control.

Many commercial systems, such as magnetic tape recording systems, use the constraint $(d,k) = (1,7)$ or $(d,k) = (2,7)$. An example of a sequence satisfying the $(d,k) = (2,7)$-RLL constraint is

$$\mathbf{w} = 00100001001000000010\,.$$

Other recording standards include the $(1,3)$-RLL constraint, which can be found in flexible disk drives, and the $(2,10)$-RLL constraint, which appears in the compact disk (CD) [Imm91, Ch. 2] and the digital versatile disk (DVD) [Imm95b].

The set of all sequences satisfying a (d,k)-RLL constraint is conveniently described by reading the binary labels of paths in the finite directed labeled graph in Figure 1.3. The graph consists of a finite collection of states (the numbered circles) and a finite collection of labeled directed edges. For each $i = 0, 1, \ldots, k-1$, there is an edge labeled 0 from state i to state $i+1$. For each $j = d, d+1, \ldots, k$, there is an edge labeled 1 from state j to state 0. A path in the graph is a sequence of edges such that the initial state of each edge is the terminal state of the preceding edge.

It can be verified that a sequence $\mathbf{w}$ satisfies the (d, k)-RLL constraint if and only if there is a path in the graph whose edge labeling is $\mathbf{w}$. For this reason, we say that the graph is a *presentation* of (or *presents*) the RLL constraint.

Any set of sequences that can be presented by a labeled graph in this way is called a *constraint* or *constrained system*. We will have much more to say about graph presentations in Chapter 2.

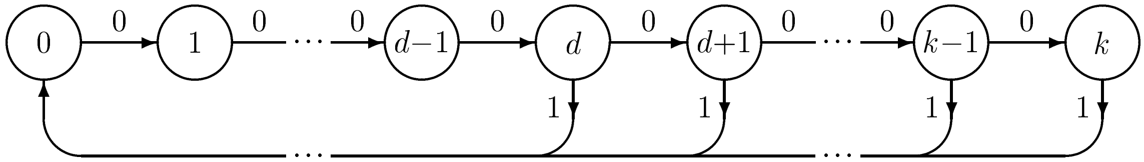

Figure 1.3: Graph presentation of the (d, k)-RLL constraint.

We may extend the class of (d, k)-RLL constraints to include the case where $k = \infty$; namely, there is no upper bound on the length of runs of 0's. In such a case, the constraint is described by the labeled graph in Figure 1.4.

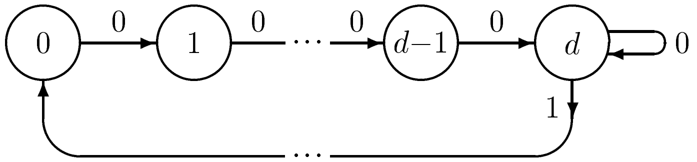

Figure 1.4: Graph presentation of the (d, ∞)-RLL constraint.

A graph presentation, such as Figure 1.3, represents a constraint by showing which sequences are allowed. An alternative representation specifies which sequences are forbidden. For instance, the $(1, 3)$-RLL constraint is the set of sequences that do not contain any word of the list

$$\{11, 0000\}$$

as a sub-string of contiguous symbols. Such a list is called a list of *forbidden words*. The reader may check that every (d, k)-RLL constraint can be defined by a very simple list of forbidden words (Problem 1.3).

1.3 Coding for Channels

In the abstract a channel can be viewed as a "black box" with inputs and outputs. The inputs represent information that is transmitted through the box. The outputs are supposed to faithfully represent the inputs. However, distortions in the channel can adversely affect the output. For this reason, coding is applied to protect the inputs.

One usually thinks of a channel as a communications system, in which information is sent from one point in space to another. Examples of communications systems include telephones, cellular phones, digital subscriber lines and deep space communications. But recording systems, such as the magnetic recording system described in Section 1.1, can also be viewed as channels. The main difference between recording channels and communications channels is that space is replaced by time. That is, in a recording channel, information is recorded at one point in time and retrieved at a later point in time [Berl80].

Current recording applications require storage devices to have very high immunity against errors. On the other hand, the ever-growing demand for storage forces the designers of such devices to write more data per unit area, thereby making the system less reliable. This is manifested in the effects of inter-symbol interference, inaccurate clocking, and random noise.

A *constrained encoder*, also known as a *modulation encoder* or *line encoder* , transforms arbitrary user data sequences into sequences, also called codewords, that satisfy a given constraint, such as an RLL constraint. Naturally, such an encoder will have a corresponding decoder called a *constrained decoder*. We loosely use the term *constrained code* to refer to the constrained encoder and decoder together. In the most general terms, the purpose of a constrained code is to improve the performance of the system by matching the characteristics of the recorded signals to those of the channel; the recorded signals are thereby constrained in such a way as to reduce the likelihood of error.

In addition to constrained coding, an error-correction code (ECC) may be used to protect the data against random noise sources. An ECC must provide an encoder (and corresponding decoder) which translates arbitrary user data sequences into codewords. A good ECC has the property that any two distinct codewords must differ enough so as to be distinguishable even after being subjected to a certain amount of channel noise. While both error-correction coding and constrained coding have been active for fifty years, the former enjoys much greater notoriety. There are many excellent textbooks on ECC, such as those by Berlekamp [Berl84], Blake and Mullin BM, Blahut [Blah83], Lin and Costello [LinCo83], MacWilliams and Sloane [MacS77], McEliece [Mcl77], Peterson and Weldon [PW72], Pless [Pl89], and Wicker [Wic95]. For an extensive summary of what is known in the area, refer to the Handbook of Coding Theory [PH98].

What is the difference between an error-correction code and a constrained code? One difference is that the "goodness" of an error-correction code is measured by how the different codewords relate to one another (e.g., in how many bit locations must any two codewords differ?), whereas the "goodness" of a constrained code is measured by properties of the individual codewords (e.g., how well does each codeword pass through the channel?).

On the other hand, this distinction is not hard and fast. Clearly, if an error-correction code is to have any value at all, then its codewords cannot be completely arbitrary and therefore must be constrained. Conversely, in recent years, there has been a great deal of interest in constrained codes that also have error-correction properties. Such developments have contributed to a blurring of the lines between these two types of coding. Nevertheless, each subject has its own emphases and fundamental problems that are shaped by the distinction posed in the preceding paragraph.

Figure 1.5 shows the arrangement of error-correction coding and constrained coding in today's recording systems. User messages are first encoded via an error-correction encoder

Figure 1.5: Coding for recording channels: standard concatenation.

and then via a constrained encoder. The resulting message is recorded on the channel. At a later time, the (possibly corrupted) version of this message is retrieved and decoded, first by the constrained decoder and then by the error-correction decoder.

This arrangement places the constrained encoder–decoder pair nearer to the channel than the error-correction encoder–decoder pair. This makes sense since otherwise the constraints imposed by the constrained encoder might be destroyed by the error-correction encoder. On the other hand, in this arrangement the ECC properties imposed by the ECC encoder may be weakened. For this reason, it may be desirable to reverse the order in which the coders are concatenated, although it is not at all obvious how to do this. These issues will be discussed in Chapter 8.

Ideally the messages recorded on a channel should be determined by a single code that has both "pairwise" error-correction properties as well as "individual" constraint properties. Such codes will be studied in Chapter 9. For now, we focus only on constrained codes.

1.4 Encoding and decoding for constrained systems

In this section, we give a rough description of the kinds of encoders and decoders that are used for constrained systems.

The encoder typically takes the form of a finite-state machine, shown schematically in

Figure 1.6. A *rate $p : q$ finite-state encoder* accepts an arbitrary input block of p user bits and generates a codeword of length q—referred to as a *q-codeword*—depending on the input block and the current internal state of the encoder. The sequences obtained by concatenating the q-codewords generated by the encoder must satisfy the constraint. There are of course only finitely many states.

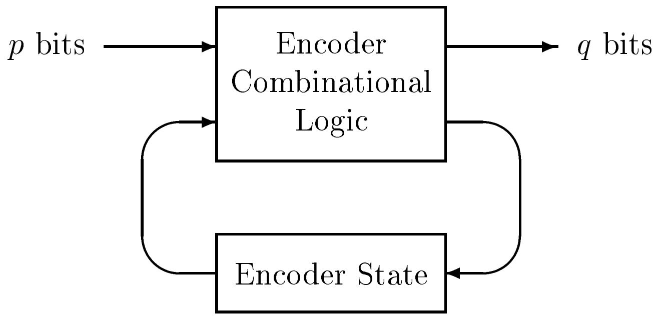

Figure 1.6: Finite-state encoder schematic.

Example 1.1 Figure 1.7 depicts a rate 2 : 3 encoder for the $(0, 1)$-RLL constrained system [Sie85a]. There are two states, A and B. Each state has exactly four outgoing edges. Each edge has two labels: a 2-bit input label and a 3-bit output label. To make a better distinction between input and output labels, we will refer to the former as input *tags*. Note that at each state, the set of four outgoing edges carry all possible 2-bit input tags exactly once.

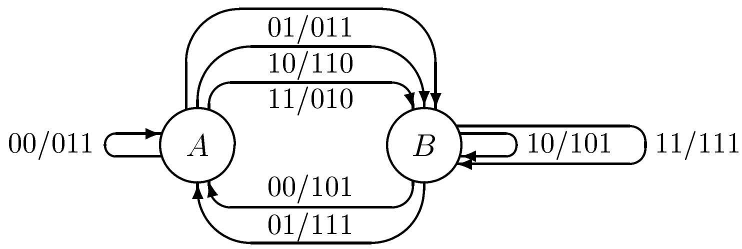

Figure 1.7: Rate 2 : 3 two-state encoder for $(0, 1)$-RLL constrained system.

One encodes as follows. First, fix an an initial state, say state A. To encode an arbitrary 2-bit input uv, first find the unique outgoing edge e, with input tag uv, from state A; the encoded 3-codeword is the output label of edge e. To encode the next 2-bit input, execute the same procedure starting from the terminal state of e. For instance, starting at state A, the sequence

$$01\ 01\ 00\ 10\ 10\ 01\ 00\ 00$$

encodes to

$$011\ 111\ 011\ 110\ 101\ 111\ 011\ 011\ ,$$

with corresponding state sequence:

$$A\ B\ A\ A\ B\ B\ A\ A\ A\ .$$

The encoded sequences do indeed satisfy the $(0,1)$-RLL constraint, i.e., one can never see two consecutive 0's; this follows from the observations: (1) one can never see two consecutive 0's on an edge output label, and (2) whenever an edge carries an output label that ends in 0, the following output labels all begin with 1. □

The encoders that we construct should be decodable. One type of decoder that we consider is a *state-dependent decoder* which accepts, as input, a q-codeword and generates a length-p block of user bits depending on the internal state, as well as finitely many upcoming q-codewords. Such a decoder will invert the encoder when applied to valid code sequences, effectively retracing the state sequence followed by the encoder in generating the codeword sequence. However, when the code is used in the context of a noisy channel, the state-dependent decoder may run into a serious problem. The noise causes errors in the detection of the codeword sequences, and the decoder must cope with erroneously detected sequences, including sequences that could not be generated by the encoder. It is generally very important that the decoder limit the propagation of errors at the decoder output resulting from such an error at the decoder input. Unfortunately, an error at the input to a state-dependent decoder can cause the decoder to lose track of the encoder state sequence, with no guarantee of recovery and with the possibility of unbounded error propagation.

Example 1.2 Consider the (admittedly artificial) rate $1:1$ encoder depicted in Figure 1.8.

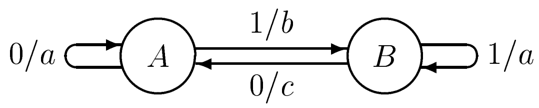

Figure 1.8: Encoder susceptible to unbounded decoder error propagation

If we select state A as the initial state, then the sequence $000000\cdots$ encodes to the sequence $aaaaaa\cdots$. If a single channel error corrupts the first a to b, then the state-dependent decoder will receive the sequence $baaaaa\cdots$ and decode to $111111\cdots$. So, a single channel error at the beginning may cause many decoding errors. □

The decoder therefore needs to have additional properties. Specifically, any symbol error at the decoder input should give rise to a limited number of bit errors at the decoder output. A *sliding-block decoder* makes a decision on a given received q-codeword on the basis of the local context of that codeword in the received sequence: the codeword itself, as well as a fixed number m of preceding codewords and a fixed number a of later codewords. Figure 1.9 shows a schematic diagram of a sliding-block decoder. It is easy to see that a single error at the input to a sliding-block decoder can only affect the decoding of codewords that fall in a "window" of length at most $\mathsf{m}+\mathsf{a}+1$ codewords.

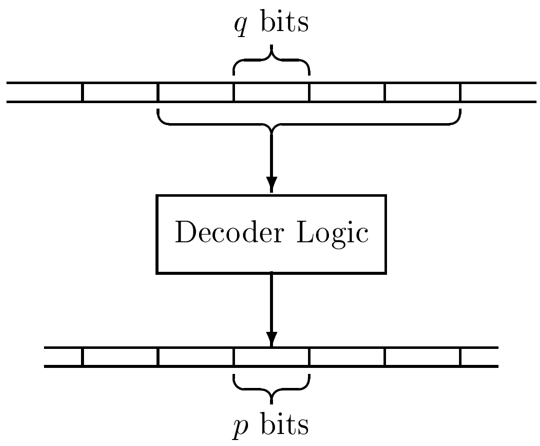

Figure 1.9: Sliding-block decoder schematic.

Example 1.3 Table 1.1 defines a sliding-block decoder for the encoder in Figure 1.7. Entries marked by "—" in the table do not affect the value of the decoded input tag. For

(current codeword)	(next codeword)	(decoded input)
010	—	11
011	101 or 111	01
011	010, 011, or 110	00
101	101 or 111	10
101	010, 011, or 110	00
110	—	10
111	101 or 111	11
111	010, 011, or 110	01

Table 1.1: Sliding-block decoder for encoder in Figure 1.7.

example, according to Table 1.1, the codeword 010 decodes to 11; the codeword 011 decodes to 01 if it is followed by 101 or 111, and it decodes to 00 if it is followed by 010, 011 or 110.

Note that this decoder really does invert the encoding in Figure 1.7. For example, the 3-codeword 010 occurs on exactly one edge as an output label. Since the corresponding input tag is 11, the codeword 010 decodes to 11. While the 3-codeword 011 occurs as an output label on two different edges, these edges end at different states. The edge ending at state B can be followed only by edges whose output labels are the codewords 101 or 111. This is why the codeword 011 decodes to 01 if it is followed by 101 or 111. Similarly, the edge ending at state A can can be followed only by edges whose output labels are the codewords 010, 011 or 110. This is why 011 decodes to 00 if it is followed by 010, 011 or 110.

Consider an application of the decoder in Table 1.1 to the following sequence of eight 3-codewords

$$011\ 111\ 011\ 110\ 101\ 111\ 011\ 011\ ;$$

this is the same sequence that was encoded in Example 1.1. It can be easily verified that the decoder will recover correctly the first seven respective 2-bit inputs

$$01\ 01\ 00\ 10\ 10\ 01\ 00\ ,$$

while the eighth 2-bit input can be either 00 or 01. Indeed, since the last 3-codeword is 011, the unique recovery of the eighth 2-bit input is possible only when the next (ninth) 3-codeword becomes available.

We point out that the inability to recover the eighth 2-bit input without additional information is not a limitation of the decoder, but rather a consequence of the structure of the encoder of Figure 1.7: the first seven 2-bit inputs lead the encoder to state A, from which there are two outgoing edges labeled 011. So, unless more information is provided (e.g., in the form of a ninth 3-codeword) there is no way to tell which edge generated the eighth 3-codeword. □

We emphasize that it is critical for both the encoder and decoder to have knowledge of the correct framing of sequences into 2-bit inputs and 3-codewords. For the former, this is a pretty safe assumption since the input comes from the well-controlled computer CPU. For the latter, this is not so safe. As discussed in Section 1.1 inaccurate clocking could cause the system to lose or gain a bit and therefore lose the framing into codewords. However, the k-constraint itself, as well as the coding/synchronization scheme to be discussed in Section 1.5.2, help to ensure that the decoder has access to the correct framing.

In the literature, the term "code rate" can refer to either the pair of numbers, $p : q$, or the ratio, p/q, between these numbers. This is a genuine ambiguity since $p : q$ is more specific than p/q. For instance, a rate 2:3 encoder is literally different from a rate 4:6 encoder. However, the two uses of the term can usually be distinguished from context. In this text we continue tradition by using the term "code rate" to refer to both $p : q$ and p/q.

Shannon proved [Sha48] that the rate p/q of a constrained encoder cannot exceed a quantity, now referred to as the Shannon capacity, that depends only upon the constraint (see Chapter 3). He also gave a non-constructive proof of the existence of codes at rates less than, but arbitrarily close to, the capacity. Therefore, as well as the issues of decodability mentioned above, any encoding/decoding scheme will be measured by the proximity of its rate, p/q, to the Shannon capacity.

Chapters 4, 5, and 6 deal with general methods of constructing encoders for constrained systems. In particular, in Chapter 5 we present an algorithm for transforming a graph presentation of any given constrained system into a finite-state encoder. Still, the design of a specific encoder for a specific constrained system usually benefits from a good deal of *ad hoc* experimentation as well.

1.5 Examples of constraints

In Sections 1.1 and 1.2, we motivated and described runlength limited constraints. In this section, we describe several other examples of constraints that arise in recording and communications applications.

1.5.1 Anti-whistle constraints

In some applications it is desirable to impose a limit on the maximum length of a particular periodic sequence. If we view a periodic sequence as a "pure tone" or "whistle," then such a constraint can be viewed as an "anti-whistle" constraint.

For example, the k-constraint limits the length of the sequence $000000\cdots$, a sequence of period 1. One may wish to the limit the length of sequences of period 2: $101010\cdots$ or $010101\cdots$, as in the following example.

Example 1.4 Figure 1.10 presents the constrained system which simultaneously satisfies

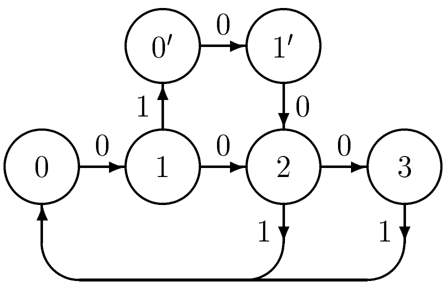

Figure 1.10: Limiting the length of the sequences $010101\cdots$ and $101010\cdots$.

the $(1,3)$-RLL constraint and limits the maximum number of repetitions of 10 or repetitions of 01 to two. □

Constraints of this type have been used in a particular version of the wireless infrared channel for data communication among mobile devices such as laptop computers. In this application, data is transmitted by infrared pulses, with a 1 in a bit cell indicating the presence of a pulse and a 0 indicating the absence of a pulse. The *duty cycle* of a sequence is simply the total number of 1's in the sequence divided by the length of the sequence. So, a sequence with a high duty cycle will transmit more pulses per unit time than a sequence with a low duty cycle. In order to conserve power, it is therefore desirable to restrict the length of transmitted sequences with high duty cycle.

As in magnetic recording, an RLL constraint is imposed in order to reduce inter-symbol interference and aid in timing control. If the constraint is a $(1, k)$-RLL constraint, then for long sequences the maximum duty cycle is roughly 50% and is achieved only by the sequences $010101\cdots$ and $101010\cdots$. Thus, it is desirable to limit the maximum length of such sequences, as in Example 1.4.

Recently, the Infrared Data Association adopted the Very Fast Infrared (IrDA-VFIr) standard which includes as part of the format a constraint that simultaneously imposes the $(1, 13)$-RLL constraint and limits the maximum number of repetitions of 10 or repetitions of 01 to five [HHH00]. Anti-whistle constraints are also used to aid in timing and gain control algorithms in magnetic recording [Berg96, p.171].

1.5.2 Synchronization constraints

Recall from the end of Section 1.2 that RLL constraints can be defined by lists of forbidden words. The example in this section is most easily defined in this way.

Example 1.5 Consider the constraint consisting of all sequences that do not contain any of the following four 24-bit words

$$010\mathbf{w}_0 01\mathbf{w}_1 \; , \quad 010\mathbf{w}_0 10\mathbf{w}_1 \; , \quad 001\mathbf{w}_0 01\mathbf{w}_1 \; , \quad 001\mathbf{w}_0 10\mathbf{w}_1 \; ,$$

where $\mathbf{w}_0 = 000000$ and $\mathbf{w}_0 = 0101010101010$. The first word consists of 01 followed by eight 0's followed by seven repetitions of 10. The second and third words each differ from the first word by a single bit shift. And the fourth word differs from the first word by two bit shifts. We leave it to the reader to construct a (finite) graph presentation of this constraint (Problem 1.12). $\square$

Such a constraint can be useful for synchronization. We describe this as follows.

Recorded data is usually grouped together into sectors consisting of a large number of bits. When a sector is addressed for retrieval, the read-head will jump to the beginning of the sector. However, it is difficult to accurately position on the first bit of the sector. Moreover, in the process of moving to the beginning of a new sector, the clock may be kicked out of synchronization. For this reason, the sector begins with special patterns that help the clock to reach synchronization quickly, to determine the beginning of encoded data and to determine the correct framing into q-codewords (if a rate $p : q$ code is used) required by a constrained decoder. One of these patterns is called a *sync mark* (see Figure 1.11).

...	Sync Mark	Encoded Data	Sync Mark	Encoded Data	Sync Mark	Encoded Data	...

Figure 1.11: Sync marks.

The position of the sync mark is determined by a correlator which examines the output of the read-head. If the sync mark is allowed to occur within the actual encoded data, then one may read a "false sync," and this could very well cause decoding errors. Thus, it is desirable to encode the data so that the sync mark is forbidden from the actual encoded data. Moreover, because of noise in the system it is also helpful to forbid the most likely perturbations of the sync mark. In one particular magnetic tape system, the first word in Example 1.5 above was used as a sync mark and the other three words were the most likely (because of bit shifts) perturbations [AJMS99]. In that system, in addition to forbidding the words in Example 1.5, a $(1, 7)$-RLL constraint as well as an anti-whistle constraint, were imposed simultaneously as well.

Finally, we mention that synchronization is also important in communications systems [Sklar88]. However, for such systems, synchronization is often dealt with using signal processing rather than constrained coding techniques.

1.5.3 Multiple-spaced runlength constraints

We next consider the class of multiple-spaced RLL constraints. These constraints are characterized by parameters (d, k, s), where d and k again define the minimum and maximum allowable runlengths of 0's, and s indicates that the runlengths of 0's must be multiples of s. In particular, $s = 2$ simply means that all runlengths of 0's must be even.

Example 1.6 A graph presentation for the $(2, 18, 2)$-RLL constraint is shown in Figure 1.12. $\square$

The $(d, k, 2)$-RLL constraints were originally investigated by Funk [Funk82] in the context of magnetic recording. More recently, and independently, $(d, k, 2)$-RLL constraints were shown to play a natural role in magneto-optic recording systems. In a particular example of such a system [HRuHC], [RuS89], not only was it detrimental to record odd run-lengths of 0's between successive 1's, but it was actually impossible! We explain this briefly as follows.

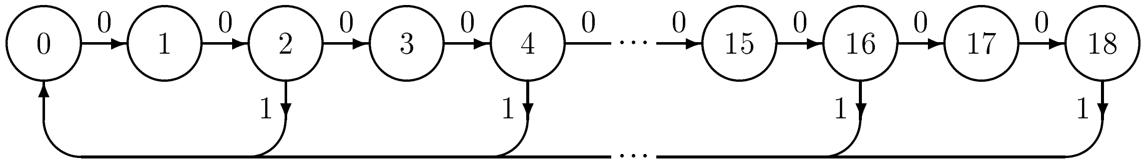

Figure 1.12: Graph presentation of the $(2, 18, 2)$-RLL constraint.

In this system, the recording medium can be magnetized only when it is heated above a certain threshold temperature. The heat is supplied by a laser, and the magnetization by a sinusoidally varying background magnetic field with constant frequency and amplitude.

The period of the field is twice the bit period T. The laser fires only at peaks (positive or negative) of the field. When it fires it marks a portion of the track with one magnetic polarity or the other (see Figure 1.13). In the figure, the cross-hatched regions indicate one

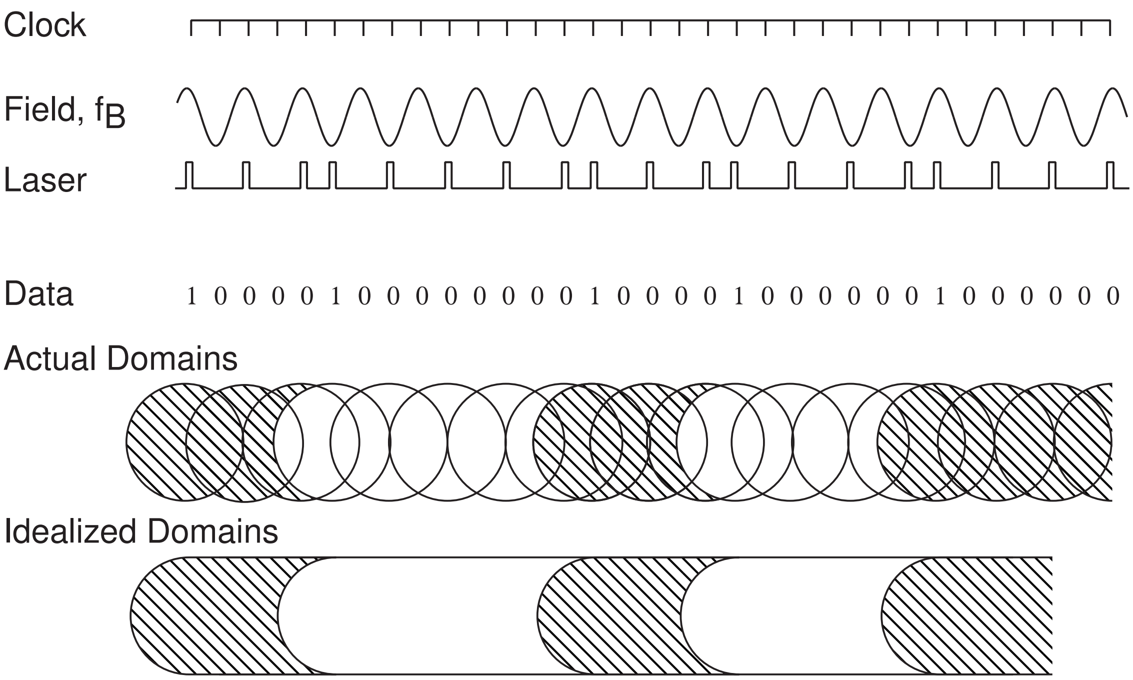

Figure 1.13: A magneto-optic recording system.

polarity and the blank regions indicate the opposite polarity. After the laser fires, it must next fire either T seconds later or $2T$ seconds later. In other words, the laser may skip one clock cycle but can never skip two consecutive clock cycles (the disk velocity is set so that there is enough overlap between marks that the track is continuously written). Thus, a magnetic transition occurs when and only when the laser does *not* skip a clock cycle. So, if a magnetic transition is caused by firing the laser at bit positions $i-1$ and i, and the next magnetic transition is caused by firing the laser at bit positions $i+k$ and $i+k+1$, then k must be even. If we view the recorded sequence as $1000\ldots0001$, where the first 1 is at bit position i and the next 1 is at bit position $i+k+1$, then the number of intervening 0's is k, an even number.

Data is read by employing the laser in a lower power mode to reflect light off the medium, instead of magnetizing the medium. The magnetic polarities, and therefore the magnetic transitions, can be determined from the reflected light by exploiting a phenomenon known as Kerr rotation. A prototype for this type of magneto-optical recording system, that employs the $(2, 18, 2)$-RLL constraint, was built by IBM [HRuHC], [RuS89], [Weig88]. For more background on optical recording, the reader is referred to [Bouw85], [Heem82], [Imm91, Ch. 2], [Pohl92].

1.5.4 Spectral-null constraints

So far, we have focused on binary sequences, mainly because runlength constraints are naturally defined on binary transition sequences. However, some constraints are naturally defined directly on the sequences of magnetic polarities that are actually recorded. We use the bipolar alphabet $\{+1, -1\}$, often denoted simply $\{+, -\}$, for these sequences, i.e., one polarity is represented by $+1$ and the opposite polarity is represented by -1. When we want to emphasize that a bipolar sequence represents a sequence of polarities, we call it a *polarity sequence*.

We say that a constrained system of bipolar sequences has a *spectral null* at a (normalized) frequency $f = m/n$ if there exists a constant B such that for all sequences $\mathbf{w} = w_0 w_1 \ldots w_{\ell-1}$ that satisfy the constrained system and $0 \le i \le i' < \ell$ we have

$$\left| \sum_{s=i}^{i'} w_s e^{-\jmath 2\pi s m/n} \right| \le B \; , \tag{1.1}$$

where $\jmath = \sqrt{-1}$ [MS87], [Pie84], [YY76] (the *actual* frequency is given by f/T, where T is the bit period). There are other equivalent ways to express this condition in terms of a power spectral density [Sklar88].

Sequences with a spectral null at $f = 0$, often called *dc-free* or *charge-constrained* sequences, have been used in many magnetic tape recording systems employing rotary-type recording heads, such as the R-DAT digital audio tape systems. Related constraints are imposed in optical recording to reduce interaction between the recorded data and the servo system, and also to allow filtering of low-frequency noise resulting from finger-prints [Imm91, Ch. 2]. The dc-free constraint is also used in communication systems, where low frequency signals tend to be distorted; this includes cable communication lines [GHW92, sec. 4.8.1] as well as fiber optic links [WF83].

When $f = 0$, the maximum value of the left-hand side in (1.1) for a given sequence $\mathbf{w} = w_0 w_1 \ldots w_{\ell-1}$ is called the *digital sum variation (DSV)* of the sequence. The DSV of $\mathbf{w}$ can also be written as

$$\max_{0 \le i \le i' < \ell} \Big| \sum_{s=i}^{i'} w_s \Big| = \Big(\max_{-1 \le r < \ell} \sum_{s=0}^{r} w_s \Big) - \Big(\min_{-1 \le r < \ell} \sum_{s=0}^{r} w_s \Big) ,$$

which is the largest difference between any two sums that are computed over prefixes of the sequence (a sum over an empty set being regarded as zero). The larger the value of B, the less reduction there will be in the spectral content at frequencies approaching the spectral null frequency $f = 0$. The set of all sequences with DSV at most B is a constrained system called the *B-charge constraint*; it is presented by the labeled graph of Figure 1.14 (see Problem 1.8).

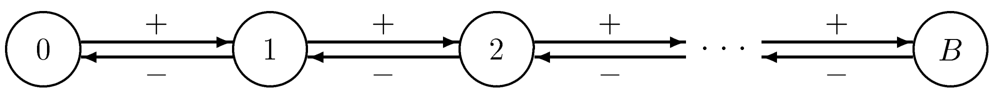

Figure 1.14: B-charge constraint.

Example 1.7 Consider the following sequence $\mathbf{w} = w_0 w_1 \ldots w_{20}$ over $\{+1, -1\}$, where we have also computed the sums $\sum_{s=0}^{i} w_s$ for $0 \le i < 21$:

w_i :	+	+	+	−	−	+	+	+	−	−	−	−	−	+	+	+	−	−	−	−	+
$\sum_{s=0}^{i} w_s$:	1	2	3	2	1	2	3	4	3	2	1	0	-1	0	1	2	1	0	-1	-2	-1

We have

$$\max_{0 \le i \le i' < 21} \Big| \sum_{s=i}^{i'} w_s \Big| = \Big(\max_{-1 \le r < 21} \sum_{s=0}^{r} w_s \Big) - \Big(\min_{-1 \le r < 21} \sum_{s=0}^{r} w_s \Big) = 4 - (-2) = 6 .$$

Therefore, the DSV of the sequence $\mathbf{w}$ is 6 (see also Figure 1.18 below). □

Graphs presenting constraints with spectral nulls at rational sub-multiples of the symbol frequency are described in [MS87]. Higher-order spectral-null constraints, which further restrict the spectral content in the vicinity of spectral-null frequencies, are discussed in [ImmB87], [KS91a], [EC91], [RSV94], [MPi89].

1.5.5 Combined charge–runlength constraints

In some applications it is desirable to impose both runlength constraints and charge-constraints. Recall that runlength constraints are expressed in the setting of binary transition sequences, and charge-constraints are expressed in the setting of bipolar polarity sequences. So, in order to describe a combined charge-runlength constraint, we must have a way of converting from one setting to the other. Formally, this is done as follows.

A bipolar polarity sequence $\mathbf{w} = w_0 w_1 w_2 \cdots$ is transformed into a binary transition sequence $\mathbf{z} = z_0 z_1 z_2 \cdots$ by the transformation

$$z_i = |w_i - w_{i-1}|/2. \tag{1.2}$$

Note that $z_i = 1$ if and only if $w_i \neq w_{i-1}$, equivalently if and only if there is a transition in magnetic polarity.

Conversely, a binary transition sequence $\mathbf{z}$ is transformed into a bipolar polarity sequence $\mathbf{w}$ through an intermediate binary sequence $\mathbf{x} = x_0 x_1 x_2 \cdots$ according to the rules

$$x_i = x_{i-1} \oplus z_i \quad \text{and} \quad w_i = (-1)^{x_i} , \tag{1.3}$$

where $\oplus$ denotes addition modulo 2. The value of x_{-1} is set arbitrarily to either 0 or 1, thereby giving rise to two sequences $\mathbf{w}$ which are the same up to overall polarity inversion. Observe that $w_i \neq w_{i-1}$ if and only if $z_i = 1$.

The transformation defined by equation (1.3) is called *precoding* because it describes how a binary sequence, such as a runlength limited sequence, must be converted into pulses of electrical current before being written on the medium. Naturally then, the transformation defined by (1.2) is called *inverse precoding.*

The B–(d,k)*-charge–RLL (CRLL) constraint* is the set of binary sequences $\mathbf{z}$ that satisfy the (d,k)-RLL constraint with the additional restriction that the corresponding precoded bipolar sequence $\mathbf{w}$ has DSV no larger than B.

Example 1.8 Consider the labeled graph in Figure 1.15. It can be verified (Problem 1.10) that any precoding of each sequence that can be generated by that graph satisfies the 6-charge constraint. Conversely, every sequence that satisfies the 6-charge constraint is a precoding of a sequence that can be generated by the graph in Figure 1.15. Therefore, the 6–$(1,3)$-CRLL constraint consists of all binary sequences that can be generated simultaneously by the labeled graphs in Figure 1.15 *and* Figure 1.16. In Section 2.4.3, we will show how, given any two labeled graphs, to construct a third labeled graph that presents the constraint defined by both graphs simultaneously. □

The 6–$(1,3)$-CRLL and 6–$(1,5)$-CRLL constraints have found application in commercial tape recording systems [Patel75], [MM77], [Mill77].

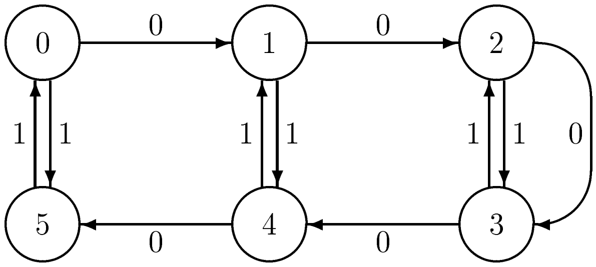

Figure 1.15: Graph presentation of precoding of the 6-charge constraint.

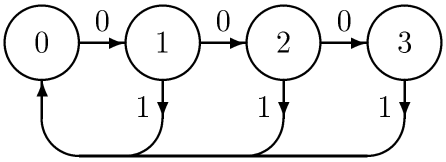

Figure 1.16: Graph presentation of the $(1,3)$-RLL constraint.

1.5.6 Constraints for PRML

Within the last decade, magnetic recording systems using digital signal processing methods have appeared on the scene. These systems typically employ a scheme, denoted PRML, based on partial-response (PR) signaling, with maximum-likelihood (ML) sequence detection. See [Cid92], [Dol89], [DMU79], [KobT70], [WoodP86]. In many recording applications, PRML has replaced peak detection.

In PRML systems it proves to be desirable to use binary sequences which satisfy not only a "global" k constraint, denoted $\mathbf{G}$, but also a separate "interleaved" k constraint, denoted $\mathbf{I}$, on the even index and odd index subsequences. The $\mathbf{G}$ constraint plays exactly the same role as the k constraint, used for timing control, as in Section 1.2. The $\mathbf{I}$ constraint aids the method (called Viterbi detection) by which data is retrieved in a PRML system. The data stream is divided into its two sub-strings (the even index and odd index), and each is retrieved separately. It turns out that an $\mathbf{I}$ constraint reduces the probability of long delay in the output of the Viterbi detector. See section 1.6.2 for more detail on PRML and the reasons for the $\mathbf{I}$ constraint.

To help distinguish the PRML $(\mathbf{G},\mathbf{I})$ constraints from the (d,k)-RLL constraints. we will use the notation $(0,\mathbf{G}/\mathbf{I})$; the 0 may be thought of as a $d=0$ constraint, emphasizing that interference between adjacent transition responses is now acceptable. An example of a sequence satisfying the $(0,4/4)$ constraint is

$$00100001001000100 1100\,.$$

We can represent $(0, \mathbf{G}/\mathbf{I})$ constraints by labeled graphs based on states which reflect the three relevant quantities, the number g of 0's since the last occurrence of 1 in the sequence and the numbers a and b which denote the number of 0's since the last 1 in the two interleaved sub-strings. We name the states by pairs (a, b), where a is the number of 0's in the interleaved sub-string containing the next to last bit, and b is the number in the sub-string containing the last bit. Note that g is a function of a and b, denoted $g(a, b)$:

$$g(a,b) = \begin{cases} 2a+1 & \text{if } a < b \\ 2b & \text{if } a \geq b \end{cases} .$$

In the (a, b) notation, the set of states V for a $(0, \mathbf{G}/\mathbf{I})$ constraint is given by

$$V = \{(a,b) \ : \ 0 \leq a, b \leq \mathbf{I} \ \text{ and } \ g(a,b) \leq \mathbf{G}\}$$

and the labeled edges between states are given by

$$\begin{array}{l} (a,b) \xrightarrow{0} (b, a+1) \ , \quad \text{provided } (b, a+1) \in V \\ (a,b) \xrightarrow{1} (b, 0) \ . \end{array}$$

Example 1.9 A graph presentation for the $(0, \mathbf{G}/\mathbf{I}) = (0, 4/4)$ constraint is shown in Figure 1.17, where state labels (omitted) agree with integer grid coordinates, starting with $(0, 0)$ at the lower left. Only the 0-labeled edges are shown in the figure; the reader can fill in the 1-labeled edges. □

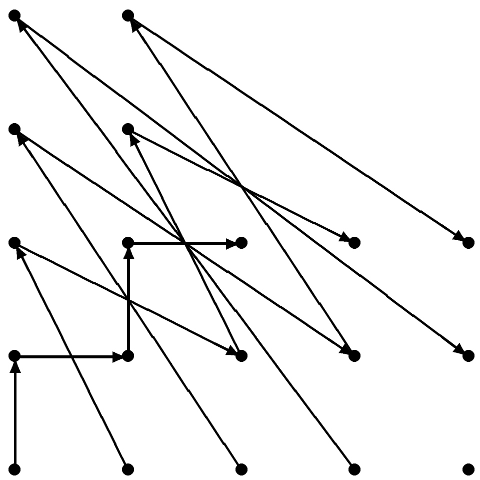

Figure 1.17: PRML $(0, \mathbf{G}/\mathbf{I}) = (0, 4/4)$ constraint: 0-labeled edges in graph presentation.

1.6 Background on magnetic recording

This section is not essential to the remainder of the text. It is intended for those readers who might be interested in understanding in a little more detail the signal processing methods used in digital magnetic recording and the motivation for introducing the constraints described in Sections 1.2 and 1.5.6.

1.6.1 Peak detection

In this section we elaborate on the description of the magnetic recording process given in Section 1.1.

Recall that in magnetic recording systems, the magnetic material at a given position along a track can be magnetized in one of two possible, opposing directions. The normalized input signal applied to the recording head in this process can be thought of as a two-level waveform $w(t)$ which assumes the values $+1$ and -1 over consecutive time intervals of bit period T. In the waveform, the transitions from one level to another, which effectively carry the digital information, are therefore constrained to occur at integer multiples of the bit period T, and we can describe the waveform digitally as a sequence $\mathbf{w} = w_0 w_1 w_2 \cdots$ over the bipolar alphabet $\{+1, -1\}$, where w_i is the signal amplitude in the time interval $(iT, (i{+}1)T]$.

Example 1.10 Figure 1.18 shows an input waveform that corresponds to the sequence $\mathbf{w}$ of Example 1.7. The figure also contains the integral of $w(t)$ over time. Since $\int_{u=0}^{(i+1)T} w(u)du = \sum_{s=0}^{i} w_s$, one sees from the figure that the DSV of $\mathbf{w}$ is indeed 6. □

Denote by $2h(t)$ the output signal (readback voltage), in the absence of noise, corresponding to a single transition from, say, -1 to $+1$ at time $t = 0$. If we assume that the input-output relationship of the digital magnetic recording channel is linear, then the output signal $y(t)$ generated by the waveform represented by the sequence $\mathbf{w}$ is given by:

$$y(t) = \sum_{i=0}^{\infty} (w_i - w_{i-1})\, h(t - iT) \ ,$$

with $w_{-1} = 1$. Note that the "derivative" sequence $\mathbf{w}'$ of coefficients $w_i' = w_i - w_{i-1}$ consists of elements taken from the ternary alphabet $\{0, \pm 2\}$, and the nonzero values, corresponding to the transitions in the input signal, alternate in sign.

A frequently used model for the transition response $h(t)$ is the function

$$h(t) = \frac{1}{1 + (2t/\tau)^2} \ ,$$

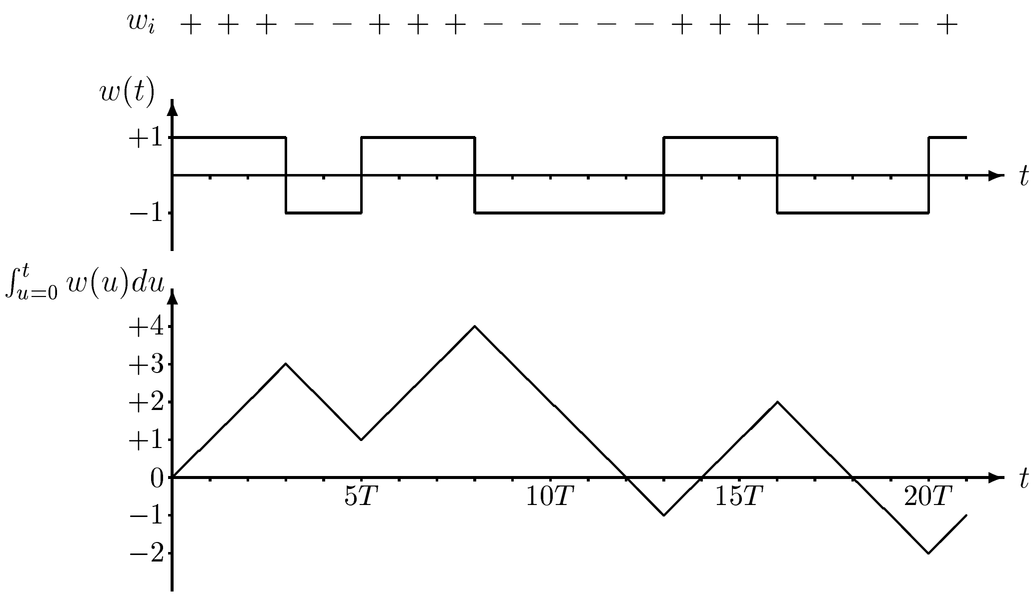

Figure 1.18: Input waveform $w(t)$ and integral $\int_{u=0}^{t} w(u)du$ that correspond to a sequence $\mathbf{w}$.

often referred to as the Lorentzian isolated-step response. The output signal $y(t)$ is therefore the linear superposition of time-shifted Lorentzian pulses with coefficients of magnitude 2 and alternating polarity. Provided that the density of transitions—reflected in the so-called density ratio τ/T—is small enough, the locations of peaks in the output signal will closely correspond to the locations of the transitions in the recorded input signal. With a synchronous clock of period T, one could then, in principle, reconstruct the ternary sequence $\mathbf{w}'$ and the recorded bipolar sequence $\mathbf{w}$.

Recall that the peak detector determines the location of peaks in the (possibly noisy) output signal whose amplitude exceeds a pre-specified level. As described earlier, runlength constraints are desirable for mitigating the effects of inter-symbol interference and improving the performance of timing recovery schemes. Specifically, the runlength constraint on binary transition sequences $\mathbf{z} = z_0 z_1 z_2 \cdots$ can be translated directly into the constraint on $\mathbf{w}'$ by means of precoding (1.3) above:

$$w_i' = w_i - w_{i-1} = (-1)^{x_{i-1}}((-1)^{z_i} - 1) = -(-1)^{x_{i-1}} \cdot 2\, z_i \ .$$

So,

$$|w_i'| = 2\, z_i,$$

and so the runlength constraints on $\mathbf{w}'$ become: the sequence $\mathbf{w}'$ contains at least d symbols and at most k symbols of value zero between successive nonzero values.

1.6.2 PRML detection

At high recording densities, the PRML (partial response maximum likelihood) approach to be discussed here has been shown to provide increased reliability relative to peak detection. The motivation for using $(0, \mathbf{G}/\mathbf{I})$ constraints can be found in the operation of the PRML system, which we now describe in simplified terms. The key difference between PRML and peak detection systems is that PRML reconstructs the recorded information from the *sequence of sample values* of the output signal at times $t = 0, T, 2T, 3T, \ldots$, rather than from individual peak locations. Denote by $\mathrm{sinc}(x)$ the real function $(\sin(\pi x))/(\pi x)$. The PRML system uses an electronic filter to transform the output pulse $2h(t)$ resulting from an isolated transition at time $t = 0$ into a modified pulse $2f(t)$ where

$$f(t) = \mathrm{sinc}\left(\frac{t}{T}\right) + \mathrm{sinc}\left(\frac{t-T}{T}\right) . \tag{1.4}$$

Note that at the consecutive sample times $t = 0$ and $t = T$, the function $f(t)$ has the value 1, while at all other times which are multiples of T, the value is 0. This particular partial-response filtering is referred to as "Class-4" [Kretz67]. Through linear superposition, the output signal $y(t)$ generated by the waveform represented by the bipolar sequence $\mathbf{w}$ is given by:

$$y(t) = \sum_{i=-1}^{\infty} (w_i - w_{i-1})\, f(t - iT) ,$$

where we set $w_{-2} = w_{-1} = w_0$. Therefore, *at sample times*, the Class-4 transition response results in *controlled interference*, leading to output signal samples $y_i = y(iT)$ that, in the absence of noise, assume values in $\{0, \pm 2\}$. Hence, in the noiseless case, the recorded bipolar sequence $\mathbf{w}$ can be recovered from the output sample values $y_i = y(iT)$, because the interference between adjacent transitions is prescribed. Therefore, unlike the peak detection system, PRML does not require the separation of transitions.

The $(0, \mathbf{G}/\mathbf{I})$ constraints arise from the following considerations. Recall that the parameter $\mathbf{G}$ is comparable to the k constraint in peak detection constraints, ensuring effective operation of the PRML timing recovery circuits, which typically rely upon frequent occurrence of nonzero output samples. Specifically, the $\mathbf{G}$ constraint represents the maximum number of consecutive zero samples allowed in the sample sequence $y_0 y_1 y_2 \cdots$.

The parameter $\mathbf{I}$ is intimately related to the maximum-likelihood detection method used in PRML. Before discussing the ML detection algorithm, it is useful to rewrite the output signal as

$$y(t) = \sum_{i=0}^{\infty} (w_i - w_{i-2})\, \mathrm{sinc}\left(\frac{t - iT}{T}\right) ,$$

where, we recall, $w_{-2} = w_{-1} = w_0$. This form can be obtained by simple arithmetic from the original expression for the Class-4 transition response (1.4). This implies the following relation between the noiseless output samples $y_0 y_1 y_2 \cdots$ and the input bipolar sequence $\mathbf{w}$:

$$y_i = w_i - w_{i-2} , \quad i \geq 0 .$$

A trellis diagram presenting the possible output sample sequences is shown in Figure 1.19. Each state is denoted by a pair of symbols that can be interpreted as the last pair of inputs, $w_{i-2}w_{i-1}$. There is an edge connecting each pair of states $w_{i-2}w_{i-1}$ and $w_{i-1}w_i$, and the label of this edge is $y_i = w_i - w_{i-2}$.

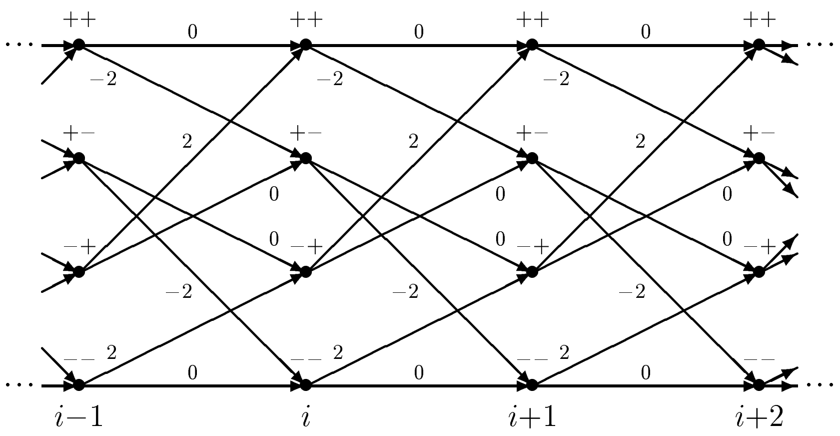

Figure 1.19: Trellis diagram for Class-4 output sequences.

The iterative ML detection algorithm is based upon the technique of dynamic programming in an embodiment of the *Viterbi algorithm*, familiar from decoding of convolutional codes. As shown by Forney [For72] and Kobayashi [Koba71], the Viterbi algorithm is an optimal detector for partial-response (in particular, Class-4) output-signal sample sequences in the presence of additive white Gaussian noise.

The behavior of the ML detector can be described in terms of the trellis diagram in Figure 1.19. Denote by $r_0 r_1 r_2 \cdots$ the sequence of (possibly noisy) received samples. Assume an initial state $u = w_{-2}w_{-1}$ is specified. For each state $v = w_{\ell-2}w_{\ell-1}$ in the diagram that can be reached from u by a path of length ℓ, the ML detector determines the allowable noiseless output sample word $\hat{y}_0\hat{y}_1 \ldots \hat{y}_{\ell-1}$, generated by a path of length ℓ from u to v, that minimizes the squared Euclidean distance

$$\sum_{i=0}^{\ell} (r_i - \hat{y}_i)^2 \ .$$

The words so determined are referred to as *survivor words.*

The representation of the output samples as $y_i = w_i - w_{i-2}$ permits the detector to operate independently on the output subsequences at even and odd time indices. Note that, within each interleave, the nonzero sample values y_i must alternate in sign. Figure 1.20 shows a trellis diagram presenting the possible sequences in each of the interleaves.

There are several formulations of the Viterbi algorithm applied to this system. See [FC89], [Koba72], [SW91], [WoodP86], [Ze87]. The motivation for the **I** constraint is clear from the following description of the ML detection algorithm, essentially due to Ferguson [Ferg72].

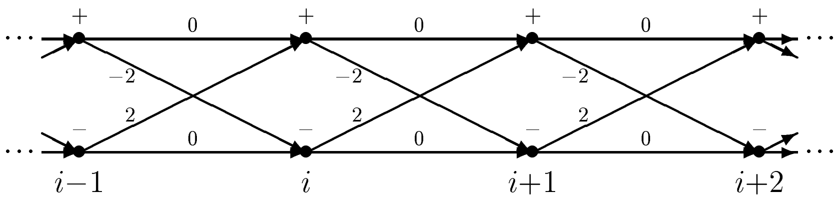

Figure 1.20: Trellis diagram for each of the interleaves.

It can be interpreted in terms of a "dynamic threshold detection scheme," as described by Wood [Wood90]. We will outline the detector operation on the subsequence of received samples at even time indices. The procedure for the odd time indices is entirely analogous.

The detector may operate in one of two modes, denoted by the variable m which takes values in $\{+1, -1\}$, according to whether the detector expects the next non-zero sample value to be positive or negative. In mode $m = +1$ (respectively, $m = -1$), the detector uses a variable R to store the *value* of the largest (respectively, smallest) sample since the last change of the mode variable m. It also maintains a variable J to store the *time index* i of the largest (respectively, smallest) sample value since the last change of the mode m.

The detector mode and variables are initialized by setting $m \leftarrow +1$, $R \leftarrow -\infty$, and $J \leftarrow -1$.

At each successive time instant $i = 2j$, $j \geq 0$, the detector takes one of three possible actions, as determined by m, R, J, and the new noisy sample r_{2j}:

1. If $m \cdot r_{2j} \geq m \cdot R$, then do $\hat{y}_J \leftarrow 0$, $R \leftarrow r_{2j}$, and $J \leftarrow 2j$;
2. else if $m \cdot R - 2 < m \cdot r_{2j} < m \cdot R$, then do $\hat{y}_{2j} \leftarrow 0$;
3. else if $m \cdot r_{2j} \leq m \cdot R - 2$, then do $\hat{y}_J \leftarrow 2m$, $R \leftarrow r_{2j}$, $J \leftarrow 2j$, and $m \leftarrow -m$.

Case 1 corresponds to the situation in which the survivor words at time $2j$ for both states in Figure 1.20 are obtained by extending the survivor word at time $2(j-1)$ for state $u = -m$. Case 2 corresponds to the situation in which the survivor word at time $2j$ for each state u is the extension of the survivor word at time $2(j-1)$ for state u. Finally, case 3 corresponds to the situation in which the survivor words at time $2j$ for both states are obtained by extending the survivor word at time $2(j-1)$ for state $u = m$.

Cases 1 and 3 correspond to "merging" of survivor paths, thereby determining the estimated value $\hat{y}_J$ for the channel output at the index of the previous merge. In the noiseless case, the merges occur when the output sample value is either $+2$ or -2. Case 2, on the other hand, defers the decision about this estimated value. In the noiseless case, this arises when the output sample value is 0. Since the latter case could arise for an arbitrary number of successive time indices, one could encounter a potentially unbounded time span between time J and the generation of the estimated channel output $\hat{y}_J$—even in the noiseless case. The $\mathbf{I}$ constraint on the maximum runlength of consecutive zero samples in each interleave of the output sequence is introduced to reduce the probability of such a long delay (or eliminate the possibility in the noiseless case).

In analogy to the RLL constraints, the $\mathbf{G}$ and $\mathbf{I}$ constraints on a binary sequence $\mathbf{z}$ translate directly to the corresponding constraints on the ideal output sample sequences $y_0 y_1 y_2 \cdots$. This is accomplished by applying precoding to each of the interleaves of $\mathbf{z}$. This *interleaved precoding* transforms $\mathbf{z}$ into a bipolar polarity sequence $\mathbf{w}$ via an intermediate binary sequence $\mathbf{x}$ according to the rules

$$x_i = x_{i-2} \oplus z_i \quad \text{and} \quad w_i = (-1)^{x_i} \; ,$$

where $x_{-2} = x_{-1} = 0$ and, as before, $\oplus$ denotes addition modulo 2. The constraint on the runlengths of consecutive 0's in the output sample sequence and in the even/odd subsequences are then reflected in corresponding $(0, \mathbf{G}/\mathbf{I})$ constraints on the binary sequences $\mathbf{z}$.

1.7 Coding in optical recording

We describe here the coding methods used in two optical-recording applications: the compact disk (CD) [Imm91, Ch. 2] and digital versatile disk (DVD) [Imm95b]. We start by a very brief and superficial description of the physical characterization of the stamped compact disk. For a much more detailed information, see [Bouw85, Ch. 7], [Heem82], [Imm91, Ch. 2], [IO85], and [Pohl92, Chapter 3].

1.7.1 The compact disk

The stamped disk consists of a round metal film that is coated by a transparent plastic material. The latter serves as a magnifying glass for the laser light that is beamed at the bottom side of the disk (i.e., the side without the label), and the reflection from the metal surface is received during readback by a light detector. Data is recorded by imprinting on the top side of the metal a sequence of pits along tracks, starting from the inner circumference to the outer (so, from the bottom side of the film, the pits seem as bumps). The pits are of varying lengths, and so are the gaps in between them. The lengths of the pits and the gaps range between $3T$ and $11T$, where $T \approx .2777\mu$m. Figure 1.21 shows portions of three adjacent tracks on the metal surface, with the lengths of the shortest and longest pits, the width of the pits, the distance between tracks (the track pitch), and the diameter of the laser spot (seen in the middle track). The lengths of the pits and gaps are determined by a $(2, 10)$-RLL constrained sequence, which is shown for the upper track ("Track i") at the top of the figure, along with a profile of the upper track as it would have appeared from the thin side of the disk, had the disk been cut along that track.

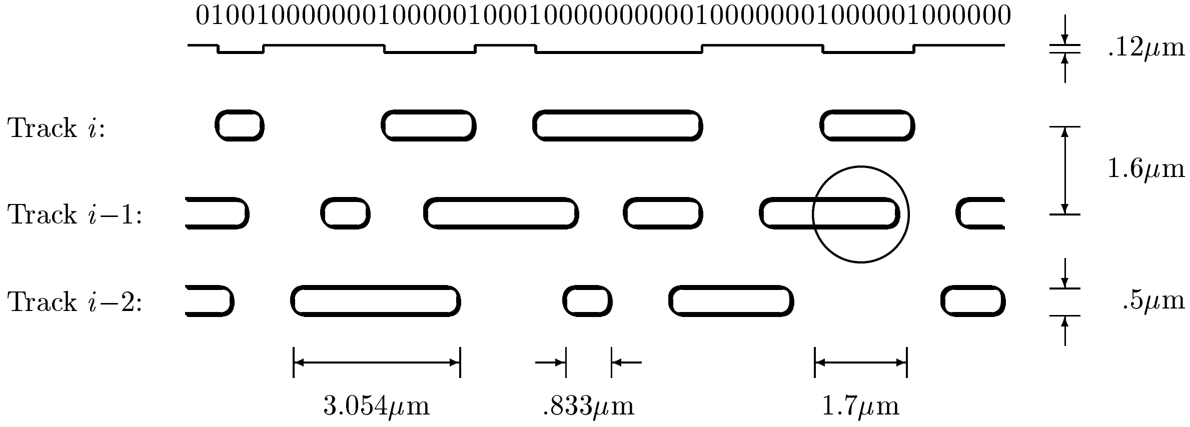

Figure 1.21: Tracks in the compact disk.

The height of the bumps (i.e, the depth of the pits) is approximately one-quarter of the wavelength of the laser beam (as the beam passes through the transparent plastic material). Therefore, a beam reflected from the bumps will destructively interfere with a reflection from the surrounding 'land.' During readback, a laser beam is aimed at a particular track and the light detector receives the reflection as the disk rotates. In this process, the bumps will appear darker to the light detector, compared to the gaps.

The parameter $T \approx .2777\mu\mathrm{m}$ has been selected so that different pit (or gap) lengths can be distinguished in the received signal. The parameters $d = 2$ and $k = 10$ have been set due to reasons akin to those that exist in magnetic recording. Specifically, pits or gaps are constrained not to be shorter than $(d+1)T = 3T$, or else the detector might have missed them (note that the diameter of the laser spot is still more than twice this shortest length). The upper bound of $(k+1)T = 11T$ is imposed to prevent any clock drift and allow clock synchronization. This requirement is particularly important in optical recording: since the linear density along a track (i.e., the parameter T) is fixed regardless of the location of the track, the angular velocity of the disk needs to be adjusted so that the *linear* velocity remains the same for all tracks. Such an adjustment is possible only if the clock can be extracted from the readback signal.

1.7.2 EFM code at rate $8 : 16$

We next describe a rate $8 : 16$ encoder for the $(2, 10)$-RLL constraint. This rate has been selected so that it matches the sub-division of the encoded bit stream into bytes.

Consider words of length 14 that satisfy the $(2,10)$-RLL constraint, with the additional property that the first and last runlengths in each word are at most 8. It turns out that there are exactly 257 words of length 14 in the $(2,10)$-RLL constraint that satisfy this property. We construct a table T that consists of 256 of those words.

A possible encoding strategy would be to encode each input byte $\mathbf{s} \in \{0,1\}^8$ into the respective entry, $\mathsf{T}[\mathbf{s}]$, in the table T. Yet, the sequence obtained by the concatenation of such words may violate the constraint. One possible solution is adding a fixed number of *merging bits* in between the generated words. Those merging bits will be used to 'glue' the words correctly.

A simple check shows that two merging bits are sufficient to guarantee that the sequence of words satisfies the $(2,10)$-RLL constraint. Table 1.2 shows an assignment of merging bits, depending on (the range of values of) the last runlength of the previously-encoded 14-bit word and (the range of values of) the first runlength of the 14-bit word that is currently encoded.

<table>
<tr><th rowspan="2">Last Encoded Runlength</th><th colspan="3">First Runlength in $\mathsf{T}[\mathbf{s}]$</th></tr>
<tr><th>0</th><th>1–7</th><th>8</th></tr>
<tr><td>0</td><td colspan="3">00</td></tr>
<tr><td>1</td><td colspan="2">00</td><td>01</td></tr>
<tr><td>2–8</td><td>00</td><td colspan="2">10</td></tr>
</table>

Table 1.2: Merging bits for a $(2,10)$-RLL encoder.

Table 1.2 thus suggests an encoding process which can be represented as a rate $8:16$ encoder with three states. Those states are named 0, 1, and 2–8, with each name standing for (the range of values that contains) the last runlength of the previously-encoded 16-bit codeword. Given an input byte $\mathbf{s} \in \{0,1\}^8$, the current codeword is obtained by preceding the entry $\mathsf{T}[\mathbf{s}]$ in T by two merging bits; those bits depend on the current encoder state and the first runlength of $\mathsf{T}[\mathbf{s}]$ according to Table 1.2. The next encoder state in then determined by the last runlength in $\mathsf{T}[\mathbf{s}]$.

We refer to the resulting encoder as a 3-*state eight-to-fourteen modulation code at rate* $8:16$, where the numbers 8 and 14 refer to the lengths of the addresses and entries, respectively, of T. This code will be denoted hereafter by 3-EFM(16), where 3 stands for the number of states and 16 is the codeword length.

The 3-EFM(16) code has a very simple sliding-block decoder that reconstructs each input byte from the corresponding 16-bit codeword, without requiring the knowledge of any previous or future codewords. Specifically, the decoder deletes the first two bits of the codeword, and the address in T of the resulting 14-bit word is the decoded input byte.

1.7.3 Dc control

While Table 1.2 lists one possible pair of merging bits for each encoder state and word in T, in certain cases there is a freedom of selection of those bits. Specifically, it can be verified that among the 257 words that can be taken as entries in T, there are 113 words in which the first runlength is between 2 and 7. When any of these words is generated from state 1 in the 3-EFM(16) code, then we could select 01 as merging bits instead of 00. In other words, we can invert the second bit of the generated codeword in this case without violating the constraint.

This freedom of inserting a bit inversion allows to reduce the digital sum variation (DSV) of the recorded sequence (see Section 1.5.4). We demonstrate this in the next example.

Example 1.11 Consider the $(2, 10)$-RLL constrained sequence

$$\mathbf{z}_1 = 010010000000100000100 ,$$

which is precoded into the signal $w_1(t)$, as shown in Figure 1.22 (refer to Section 1.5.5 to the precoding rule (1.3), and also to Example 1.7 and Figure 1.18). The value of the integral $\int_{u=0}^{t} w_1(u)du$ ranges between -1 and $+7$ and, therefore, the DSV of the recorded signal equals 8.

Suppose now that the eighth bit in $\mathbf{z}_1$ is inverted to form the sequence

$$\mathbf{z}_2 = 010010010000100000100 .$$

Denoting by $w_2(t)$ the respective recorded signal, the integral $\int_{u=0}^{t} w_2(u)du$ now ranges between -3 and $+3$, thereby reducing to DSV to 6. $\square$

As explained earlier, the recording in optical media is based on changing the reflectivity of the recording surface. During readback, the surface is illuminated by a laser beam and the readback signal is obtained by measuring the amplitude of the reflection beam: high reflection ('bright surface') means a value $+1$, while low reflection ('dark surface') stands for -1. To determine whether the reflection is high or low, the detector needs to be calibrated to the zero level (referred to as the 'dc,' or 'direct current' level) of the signal. To this end, the recorded signal is constrained to have an average value close to zero over a given window length. This, in turn, allows the required calibration simply by computing the average of the amplitude of the reflected beam over a given period of time.

In Section 1.5.5, we introduced the combined charge-runlength constraints. If a B–$(2, 10)$-CRLL encoder were used for some value B, then, certainly, we would have an average value close to zero for every *individual* constrained sequence generated by the encoder. However, a small value for B would force the encoding ratio to drop below $8/16 = .5$, while a large value of B would result in a too complex encoder.

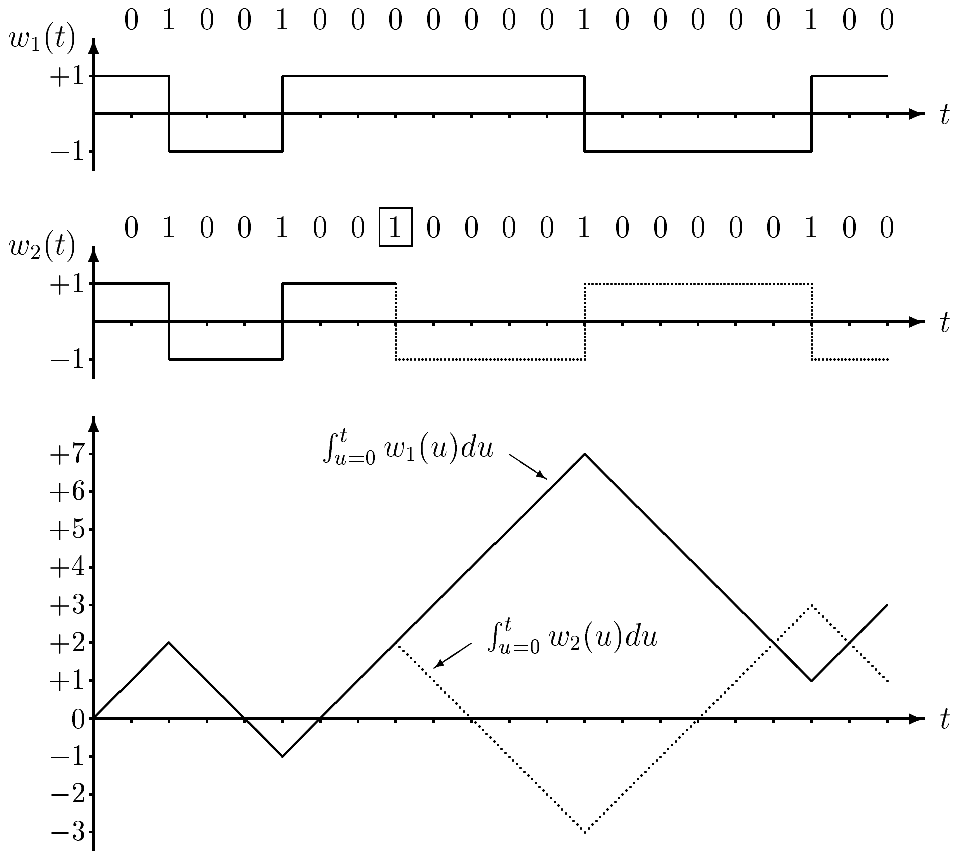

Figure 1.22: Effect of inverting a bit on DSV.

On the other hand, for the purpose of training on the zero level of the signal, it would be possible to weaken the charge constraint so that the DSV would be bounded over a sufficiently long sequence with very high probability (yet not necessarily for every individual sequence), where the probability is computed assuming (say) a uniform distribution over the sequences of input bytes. Indeed, the freedom of selecting the merging bits provides such a *dc (or DSV) control*: we will show in Section 4.7 (Example 4.12) that under a uniform distribution on the input bytes, once in approximately every ten input bytes (on average), we will be at state 1 with the freedom of selecting either 00 and 01 as merging bits

The freedom of selecting the merging bits can be increased by refining the definition of the states in the 3-EFM(16) code and designating a separate state for each possible value of the last runlength in a codeword. That is, instead of having only three states, 0, 1, and 2–8, we will have nine states that will be denoted 0 through 8; so, an input byte $\mathbf{s}$ will lead the encoder to state i, where i is the last runlength of $\mathsf{T}[\mathbf{s}]$. The resulting encoder will be called a 9-EFM(16) code.

To see the advantage of the 9-EFM(16) code, consider the following example. Suppose that while at state 2–8 in the 3-EFM(16) code, the input byte $\mathbf{s}$ is such that $\mathsf{T}[\mathbf{s}]$ starts with a runlength 1. By Table 1.2, the merging bits in this case are 10, and, since it is possible to enter state 2–8 after generating a codeword that ends with a runlength 8, we cannot allow any other values for the merging bits in this case. On the other hand, if each of the runlength values 2 through 8 leads to a separate state, then, while at any of the states 2 through 7, the merging bits 00 are also admissible for the mentioned byte $\mathbf{s}$.

Clearly, the advantage of the 9-EFM(16) code comes with a price: this code is somewhat more complex than the 3-EFM(16) code. Table 1.3 presents the possible selection of merging bits for each state in the 9-EFM(16) code and each first runlength in $\mathsf{T}[\mathbf{s}]$.

<table>
<tr><th></th><th colspan="9">First Runlength in $\mathsf{T}[\mathbf{s}]$</th></tr>
<tr><th>State</th><th>0</th><th>1</th><th>2</th><th>3</th><th>4</th><th>5</th><th>6</th><th>7</th><th>8</th></tr>
<tr><td>0</td><td colspan="9">00</td></tr>
<tr><td>1</td><td colspan="2">00</td><td colspan="6">00, 01</td><td>01</td></tr>
<tr><td>2</td><td>00</td><td>00, 10</td><td colspan="5">00, 01, 10</td><td colspan="2">01, 10</td></tr>
<tr><td>3</td><td>00</td><td>00, 10</td><td colspan="4">00, 01, 10</td><td colspan="3">01, 10</td></tr>
<tr><td>4</td><td>00</td><td>00, 10</td><td colspan="3">00, 01, 10</td><td colspan="4">01, 10</td></tr>
<tr><td>5</td><td>00</td><td>00, 10</td><td colspan="2">00, 01, 10</td><td colspan="5">01, 10</td></tr>
<tr><td>6</td><td>00</td><td>00, 10</td><td>00,01,10</td><td colspan="6">01, 10</td></tr>
<tr><td>7</td><td>00</td><td>00, 10</td><td colspan="7">01, 10</td></tr>
<tr><td>8</td><td>00</td><td>10</td><td colspan="7">01, 10</td></tr>
</table>

Table 1.3: Admissible merging bits in the 9-EFM(16) code.

While the freedom of selecting the merging bits in the 9-EFM(16) code could be sufficient for calibrating the zero level of the readback signal, charge constraints are required in optical recording also for the suppression of the power spectral density of the recorded signal at the low-frequency range. Such a suppression is necessary so that low-frequency control signals and noise can be filtered out without distorting the recorded signal (see [Imm91, p. 27]).

It turns out that the dc control that is attainable by the 9-EFM(16) code does not provide sufficient reduction of the power spectral density at the low-frequency range. This was resolved in the compact disk by inserting three merging bits instead of two, resulting in the (proper) EFM code , at rate 8 : 17, which we denote by EFM(17) [Imm99, Section 14.4.1].

The solution in the DVD was different. Here, the encoding is not based on merging bits; instead, the encoder has four states, and each state has its own encoding table that maps input bytes into 16-bits codewords. In addition, 88 out of the 256 input bytes can be encoded at each state into two possible codewords, thereby introducing freedom that allows sufficient suppression of the low-frequency range of the power spectral density (see [Imm95b] and [Imm99, Section 14.4.2]). The resulting encoder, called the EFMPlus code, has rate 8 : 16. Yet, the decoding of an input byte requires the knowledge of *two* consecutive 16-bit codewords, compared to the EFM(16) and the EFM(17) codes where one codeword is sufficient. Alternative coding schemes at rate 8 : 16 have been suggested where the desired dc control can be achieved with a decoding window length of one codeword only; see [Roth00] and [Imm99, Section 14.4.4].

1.8 Two-dimensional constraints

Constrained systems, as we have defined them so far, are sets of sequences; each sequence can be viewed as a one-dimensional stream of symbols in a one-dimensional space. In an analogous way, one can define constraints on two-dimensional (or even higher-dimensional) arrays.

Example 1.12 Consider the set of all two-dimensional arrays that satisfy the following condition: whenever one sees a 0 in the array, at least one of its four neighbors (up, down, left or right) is also a 0; in other words, the pattern:

$$\begin{array}{ccc} & 1 & \\ 1 & 0 & 1 \\ & 1 & \end{array} \tag{1.5}$$

is forbidden. □

One application of this kind of constraint occurs in holographic recoding systems, as shown in Figure 1.23 [HBH94]. In such a system, a laser illuminates a programmable array called a spatial light modulator (SLM). The SLM is an array of shutters, each of which can be open or closed; the open shutters allow light to pass through, while the closed shutters prevent light from passing through. So, if a two-dimensional array of 0's and 1's is programmed onto the SLM, illumination of the SLM by the laser will create an optical representation, called the *object beam*, of the array. The object beam is then focused through a lens. A *reference beam*, which is a simple plane wave propagating at some angle with respect to the medium, interferes with the focused object beam at a spot on the recording medium and writes a pattern (the hologram).

The object beam can be reproduced at a later time by illuminating the medium with the same reference beam that was used to record it; light from the reproduced object beam is passed through a lens and then collected on a photosensitive array of detectors, known as a charge coupled device (CCD). In this way, the original binary array of data can be recovered.

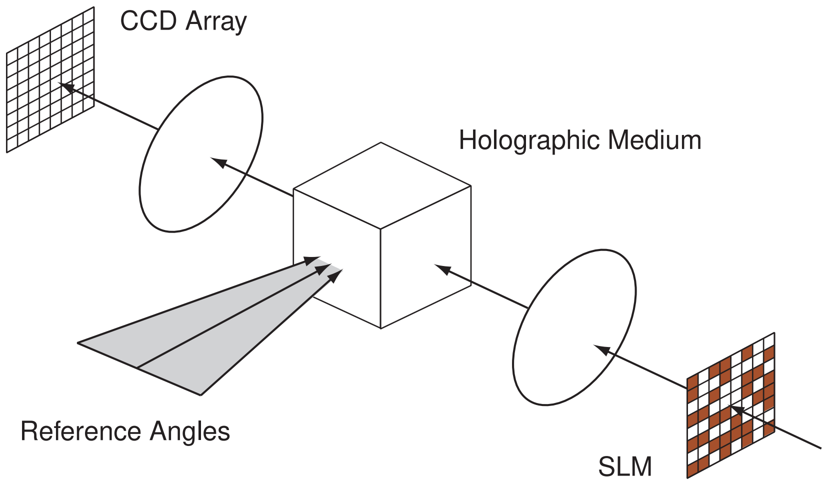

Figure 1.23: A holographic data storage system (figure taken from [HBH94]).

By varying the angle of propagation of the reference beam, several holograms can be recorded in the same physical spot, in a scheme known as angular multiplexing. Since many holograms can be recorded in one spot, holographic data storage holds the promise of very high data density. Since an entire array of data can be retrieved all at once, it also holds the promise of very high data rate.

Now, inter-pixel interference (IPI), the two-dimensional analog of inter-symbol interference, tends to occur when a 0 is surrounded by 1's; specifically, IPI may be provoked by the pattern (1.5) above. Since IPI degrades the performance of a holographic recording system, it may be desirable to forbid such a pattern. The collection of $N \times N$ arrays which do not contain the pattern (1.5) is a two-dimensional constrained system of arrays. It then becomes important to find methods of encoding arbitrary binary sequences into such a constrained system.

Recently, there has been a great deal of interest in coding for two-dimensional constrained systems. While some of this has been specifically geared towards applications such as holographic recording (e.g., [AM98], [BM99], [KN99]) a great deal of it has been focused on efforts to extend the general one-dimensional theory of constrained systems, as presented in this text, to two dimensions (e.g., [KZ98], [RSW00]). At this point, the two-dimensional theory is still sketchy. For instance, while we will see in Chapter 3 that there is an explicit formula for the capacity of any one-dimensional constrained system, there is nothing like this known in two dimensions.

Problems

Problem 1.1 Any sequence $\mathbf{z}$ that satisfies the (d, k)-RLL constraint begins with a certain number, j, of 0's, where $0 \le j \le k$. As a function of j, identify the states in Figure 1.3 from which $\mathbf{z}$ can be generated.

Problem 1.2 Show that a sequence satisfies the $(0, 1)$-RLL constraint if and only if the inverted sequence (obtained by changing each 0 to 1 and each 1 to 0) satisfies the $(1, \infty)$-RLL constraint.

Problem 1.3 For given d and k, exhibit a list of exactly $d+1$ words that constitute a forbidden list for the (d, k)-RLL constraint. Is $d + 1$ the minimal size of such a list?

Problem 1.4 A binary sequence satisfies the asymmetric (d_0, k_0, d_1, k_1)-RLL constraint if it consists of alternating runs of 0's and 1's such that the runlengths of 0's lie in between d_0 and k_0 and the runlengths of 1's lie in between d_1 and k_1. Draw a graph presentation of the $(1, 3, 2, 5)$-RLL constraint.

Problem 1.5 Change the assignment of input tags on the encoder in Figure 1.8 so that it has a corresponding sliding-block decoder.

Problem 1.6 For a given (d, k)-RLL constraint and a given integer N, identify the sequences of length N that satisfy the constraint and have maximal duty cycle (i.e., maximal percentage of 1's).

Problem 1.7 Verify that the transformations, precoding (1.3) and inverse precoding (1.2), are inverses of one another in the following sense:

1. the inverse precoding of the precoding of $\mathbf{z}$ is $\mathbf{z}$;
2. the precoding of the inverse precoding of $\mathbf{w}$ is either $\mathbf{w}$ or the sequence obtained from $\mathbf{w}$ by interchanging 1's and -1's.

Problem 1.8 Let $\mathbf{w} = w_0 w_1 w_2 \cdots$ be a sequence over the alphabet $\{+1, -1\}$. Show that $\mathbf{w}$ can be generated by the labeled graph in Figure 1.14 starting at state j if and only if

$$- \min_{-1 \le r < \ell} \sum_{s=0}^{r} w_s \le j \le B - \max_{-1 \le r < \ell} \sum_{s=0}^{r} w_s .$$

Deduce that $\mathbf{w}$ satisfies the B-charge constraint if and only if it can be generated by the labeled graph in Figure 1.14.

Problem 1.9 Show that Figure 1.24 generates all the sequences obtained by precoding, given by formula (1.3), of sequences that satisfy the $(1, 3)$-RLL constraint. Draw the corresponding picture for arbitrary d and k.

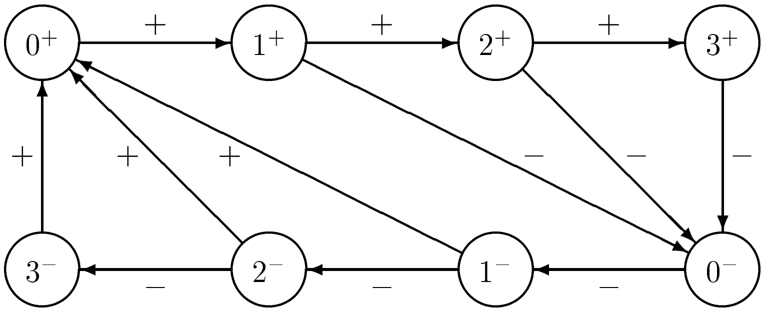

Figure 1.24: Graph presentation of precoding of the $(1, 3)$-RLL constraint.

Problem 1.10 Show that a sequence satisfies the 6-charge constraint if and only if it can be obtained by precoding of a sequence generated by the labeled graph in Figure 1.15.

Problem 1.11 Fill in the edges labeled by 1 in the graph presentation Figure 1.17.

Problem 1.12 Construct a (finite) graph presentation of the constraint described in Example 1.5.

Problem 1.13 Let ℓ be a positive integer and let $Z_{2\ell}$ denote the set of all sequences of length 2ℓ that satisfy the $(1, \infty)$-RLL constraint. Define the mapping $f : Z_{2\ell} \to \{+, -\}^{2\ell}$ as follows: $f(z_0, z_1, \ldots, z_{2\ell-1}) = w_0 w_1 \ldots w_{2\ell-1}$, where for each $i = 0, 1, \ldots, \ell-1$, the values of w_{2i} and w_{2i+1} depend on the values of z_{2i-1}, z_{2i}, and z_{2i+1} according to Table 1.4 (we assume that $z_{-1} = 0$).

z_{2i-1}	z_{2i}	z_{2i+1}	w_{2i}	w_{2i+1}
0	0	0	$-$	$+$
0	0	1	$+$	$+$
0	1	0	$+$	$-$
1	0	0	$-$	$-$
1	0	1	$+$	$-$

Table 1.4: Mapping for Problem 1.13.

1. Show that the mapping f is one-to-one.
2. Let $w_0 w_1 \ldots w_{2\ell-1}$ be an image of $z_0 z_1 \ldots z_{2\ell-1}$ under f. Show that for each $i = 0, 1, \ldots, \ell-1$,
$$\sum_{s=0}^{2i+1} w_s = 2z_{2i+1} .$$
3. Show that the images of f satisfy the 2-charge constraint.

4. Let $\mathbf{w} = w_0 w_1 \ldots w_{2\ell-1}$ be a sequence that satisfies the 2-charge constraint. Show that $\mathbf{w}$ is an image under f if one of the following conditions holds:

   (a) $w_{2i} \neq w_{2i+1}$ for every $0 \leq i < \ell$, or else —

   (b) if i is the first index such that $w_{2i} = w_{2i+1}$, then $w_{2i} = w_{2i+1} = +$.

5. Deduce that the number of sequences of length 2ℓ that satisfy the 2-charge constraint is at least—but no more than twice—the size of $Z_{2\ell}$.

Problem 1.14 (Enumeration of (d,k)-RLL sequences) Fix d and k to be integers $0 \leq d \leq k < \infty$, and denote by $x(\ell)$ the number of words of length ℓ that satisfy the (d,k)-RLL constraint. Also, denote by $x_0(\ell)$ the number of such words that do not contain any 1's among their first d bits. Define $x(\ell) = x_0(\ell) = 0$ for $\ell < 0$ and $x(0) = x_0(0) = 1$.

1. Show that $x_0(\ell)$ satisfies for $\ell \geq 0$ the recurrence

$$x_0(\ell) = u(k{-}\ell) + \sum_{i=d+1}^{k+1} x_0(\ell{-}i) ,$$

   where

$$u(t) = \begin{cases} 0 & \text{if } t < 0 \\ 1 & \text{if } t \geq 0 \end{cases} .$$

2. Show that for every ℓ,

$$x(\ell) = \sum_{i=0}^{d} x_0(\ell{-}i) .$$

3. Conclude from parts 1 and 2 that $x(\ell)$ satisfies for $\ell > k$ the recurrence

$$x(\ell) = r(k{+}d{+}1{-}\ell) + \sum_{i=d+1}^{k+1} x(\ell{-}i) ,$$

   where

$$r(t) = \begin{cases} 0 & \text{if } t < 0 \\ t & \text{if } t \geq 0 \end{cases} .$$

   (Note that this recurrence becomes linear when $\ell > k{+}d$.)

4. Show that the values of $x(\ell)$ for $0 \leq \ell \leq k$ satisfy

$$x(\ell) = \begin{cases} \ell{+}1 & \text{for } 0 \leq \ell \leq d \\ x(\ell{-}1) + x(\ell{-}d{-}1) & \text{for } d < \ell \leq k \end{cases} .$$

5. Assume now that $k = \infty$. Show that for $\ell > d$,

$$x_0(\ell) = x_0(\ell{-}1) + x_0(\ell{-}d{-}1) ,$$

   with the initial conditions

$$x_0(\ell) = 1 , \quad 0 \leq \ell \leq d .$$

6. Show that when $k = \infty$, the values $x(\ell)$ are related to $x_0(\ell)$ by

$$x(\ell) = x_0(\ell+d) \ ;$$

so, $x(\ell)$ satisfies for $\ell > d$,

$$x(\ell) = x(\ell-1) + x(\ell-d-1) \ ,$$

with the initial conditions

$$x(\ell) = \ell+1 \ , \quad 0 \le \ell \le d \ .$$