Coding techniques for the optical and magnetic recording channel

Citation for published version (APA):

Document status and date:
Published: 01/01/1990

Document Version:
Publisher’s PDF, also known as Version of Record (includes final page, issue and volume numbers)

Please check the document version of this publication:

• A submitted manuscript is the version of the article upon submission and before peer-review. There can be important differences between the submitted version and the official published version of record. People interested in the research are advised to contact the author for the final version of the publication, or visit the DOI to the publisher’s website.
• The final author version and the galley proof are versions of the publication after peer review.
• The final published version features the final layout of the paper including the volume, issue and page numbers.

Link to publication

General rights
Copyright and moral rights for the publications made accessible in the public portal are retained by the authors and/or other copyright owners and it is a condition of accessing publications that users recognise and abide by the legal requirements associated with these rights.

• Users may download and print one copy of any publication from the public portal for the purpose of private study or research.
• You may not further distribute the material or use it for any profit-making activity or commercial gain
• You may freely distribute the URL identifying the publication in the public portal.

If the publication is distributed under the terms of Article 25fa of the Dutch Copyright Act, indicated by the “Taverne” license above, please follow below link for the End User Agreement:
www.tue.nl/taverne

Take down policy
If you believe that this document breaches copyright please contact us at:
openaccess@tue.nl
providing details and we will investigate your claim.

Download date: 23. Dec. 2019
Survey Course:
Nr. 274/1 GRADUATE SEMESTER 1990

Subject : DIGITAL AUDIO SYSTEMS
Lecturer : Dr. Ir. K.A. SCHOUHAMER IMMINK
Surveyed by : Dr. Ir. K.A. Schouhamer Immink
Copy : Mr. Sallam
Coding Techniques for the Optical and Magnetic Recording Channel

Kees A. Schouhamer Immink
Philips Research Laboratories
Eindhoven
CONTENTS

1 Compact Disc: A Design Case ........................................ 3
  1.1 Introduction ......................................................... 3
  1.2 Channel Description ................................................ 6
  1.3 Channel Coding ...................................................... 9
  1.4 Further Requirements for the Recording Code .................... 14
  1.5 EFM Recording Code ................................................. 15

2 Entropy and Capacity .................................................. 20
  2.1 Introduction ........................................................... 20
  2.2 Information Content of a Message, Entropy ....................... 21
    2.2.1 Entropy of Memoryless Sources .............................. 21
    2.2.2 Markov Chains .................................................. 23
    2.2.3 Entropy of Markov Information Sources ..................... 28
  2.3 Capacity of Discrete Noiseless Channels ........................ 30
    2.3.1 Capacity of Markov Information Sources .................... 30
    2.3.2 Variable-Length Symbols ...................................... 34
    2.3.3 Interleaved Sequences ........................................ 36
  2.4 References .......................................................... 36

3 Runlength-Limited Sequences ......................................... 37
  3.1 Introduction ........................................................... 37
  3.2 Number of Binary (d,k) Sequences Versus Sequence Length n .. 39
  3.3 Asymptotic Information Rate ....................................... 41
    3.3.1 State-Transition Matrix Description ......................... 45
  3.4 Statistical Properties of Maxentropic Runlength-Limited Sequences .. 46
    3.4.1 Spectrum of Maxentropic RLL Sequences ..................... 48
  3.5 Practical Coding Schemes .......................................... 50
    3.5.1 Fixed-Length Binary (d,k) Codes ............................ 50
Compact Disc: A Design Case

The Compact Disc system can be considered as a transmission system that brings sound from the studio into the living room. The sound encoded into data bits and modulated into channel bits is sent along the ‘transmission channel’ consisting of write laser - master disc - user disc - optical pick-up. In this chapter, we shall provide a heuristic description of the various factors that play a role in the design of the Compact Disc recording code EFM.

1.1 Introduction

In this chapter, we shall deal in some detail with the various factors that had to be weighed one against the other in the design of the Compact Disc system. In particular, we shall discuss the EFM recording code (‘Eight-to-Fourteen Modulation’), which helps to produce the desired high information density on the disc.

With its high information density and a playing time of 70 minutes, the outside diameter of the disc is only 120 mm. Because the disc is so compact, the dimensions of the player can also be small. The way in which the digital information is derived from the analog sound signal gives a frequency characteristic that is flat from 20 to 20000 Hz. With this system the well-known wow and flutter of conventional players are a thing of the past.

In the Compact Disc system, the analog audio signal is sampled at a rate of 44.1 kHz, which is, according to Nyquist’s sampling theorem, sufficient for reproduction of the maximum audio frequency of 20 kHz. The signal is quantized by the method of uniform quantization: the sampled amplitude is divided into equal parts. The number of bits per sample is 32, that is 16 for the left and 16 for the right audio channel. This corresponds to a signal-to-noise ratio of more than 90 dB. The net bit rate is thus $44100 \times 32 = 1.41 \text{ Mbit/s}$. The audio bits are grouped
into blocks of information, called *frames*, each containing six of the original samples.

Successive blocks of audio bits have blocks of parity symbols added to them in accordance with a coding technique called *CIRC* (Cross-Interleaved Reed-Solomon Code). The ratio of the number of bits before and after this operation, the rate of the CIRC code, is 3:4. The parity symbols can be used here to correct errors, or just to detect errors if correction is found to be impossible. These errors may stem from defects in the manufacturing process, for example, undesired particles or air bubbles in the plastic substrate, or damage during use, or fingermarks or dust on the disc. Since the information with the CIRC code is 'interleaved' in time, errors that occur at the input of the error-correction system are spread over a large number of frames during decoding. A flaw such as a scratch can often produce a train of errors, called an *error burst*. The built-in error-correction system can correct a burst of up to 4000 data bits, largely because the errors are spread out by interleaving. If more errors than the permitted maximum occur, they can only be detected; the errors detected can be masked.

Each frame then has subcode C&D (Control and Display) bits added to it; one of the functions of the C&D bits is providing the 'information for the listener'. In some of the versions of the player the information for the listener can be represented on a display and the different sections of the music can be played in the order selected by the user. After the previous operation the bits are called *data bits*.

Next the bit stream is encoded, that is to say the data bits are translated into channel bits, which are suitable for storage on the disc. The EFM code is used for this: in EFM blocks of eight bits are translated into blocks of fourteen bits. The blocks of fourteen bits are linked by three *merging bits*. The ratio of the number of bits before and after the EFM encoding stage is 8:17.

For the synchronization of the bit stream an identical synchronization pattern consisting of 27 channel bits is added to each frame. The total bit rate after all these manipulations is $4.32 \times 10^6$ channel bits/s. Table 1.1 gives a survey of the successive operations with the associated bit rates, and their names. From the magnitude of the channel bit rate and the scanning velocity 1.25 m/s it follows that the length of a channel bit on the disc is approximately 0.3 μm.

Figure 1.1 represents the complete Compact Disc system as a 'transmission system' that brings the sound from the studio into the living room. The orchestral sound is converted at the recording end into a bit stream $B_i$, which is recorded on the master disc. The master disc is
TABLE 1.1 - Names of the successive signals, the associated bit rates and operations during the processing of the audio signal.

<table>
<thead>
<tr>
<th>Name</th>
<th>Bit rate ($10^6$ bits/s)</th>
<th>Operations</th>
</tr>
</thead>
<tbody>
<tr>
<td>Audio signal</td>
<td></td>
<td>PCM (44.1 kHz)</td>
</tr>
<tr>
<td>Audio bit stream</td>
<td>1.41</td>
<td>CIRC (+parity bits)</td>
</tr>
<tr>
<td>Data bit stream</td>
<td>1.94</td>
<td>EFM (+sync. bits + C&amp;D)</td>
</tr>
<tr>
<td>Channel bit stream</td>
<td>4.32</td>
<td></td>
</tr>
</tbody>
</table>

Figure 1.1 - Compact Disc system considered as a transmission system that brings sound from the studio into the living room. The transmission channel between the encoding system (COD) at the recording side and the decoding system (DECOD) in the player 'transmits' the bit stream $B_i$ to DECOD via the write laser, the master disc (MD), the disc manufacture, the disc (D) in the player and the optical pick-up; in the ideal case $B_n$ is the same as $B_i$. The bits of $B_n$, as well as the clock signal (Cl) for further digital operations, have to be detected from the output signal of the pick-up unit at Q.

used as the 'pattern' for making the discs for the user. The player in the living room derives the bit stream $B_0$ - which in the ideal case should be a facsimile of $B_i$ - from the user disc and reconverts it to the orchestral sound. The system between COD and DECOD is the actual transmission channel: $B_i$ and $B_0$ consist of channel bits.

Figure 1.2 shows the encoding system in more detail. The audio signal is first converted into a stream $B_1$ of audio bits by means of pulse-code modulation (PCM). The parity bits for error correction and a number of bits for 'control and display' (C&D) are then added to the bit stream. This results in the data bit stream $B_2$. The modulator converts this into channel bits ($B_3$). The bit stream $B_i$ is obtained by adding a synchronization signal.
FIGURE 1.2 - Encoding system (COD in Figure 1.1). The system is highly simplified here; in practice there are two channels for stereo which together supply the bit stream \( B_i \). The bit stream \( B_1 \) is supplemented by parity and C&D bits (\( B_2 \)) translated by the encoder (\( B_3 \)) and provided with synchronization signals (\( B_i \)). MUX: multiplexers.

The number of data bits \( N_b \) that can be stored on the disc is given by

\[
N_b = \eta A/d^2,
\]

where \( A \) is the useful area of the disc surface, \( d \) is the diameter of the laser light spot on the disc and \( \eta \) is the 'number of data bits per spot' (the number of data bits that can be resolved per length \( d \) of track). \( A/d^2 \) is the number of spots that can be accommodated side by side on the disc. The information density \( N_b/A \) is thus given by

\[
N_b/A = \eta/d^2. \tag{1.1}
\]

The spot diameter \( d \) is one of the most important parameters of the channel. The channel coding can give a higher value of \( \eta \) and thus of \( N_b \). We shall now briefly explain some of the aspects of the channel that govern the specification for the recording code.

### 1.2 Channel Description

The bit stream \( B_i \) in Figure 1.1 is converted into a signal at \( P \) that switches the light beam from the write laser on and off. The channel should be of high enough quality to allow the bit stream \( B_i \) to be reconstituted from the read signal at \( Q \). To achieve this quality, all the stages in the transmission path must meet exacting requirements, from
the recording on the master disc, through the disc manufacture, to the actual playing of the disc. The quality of the channel is determined by the player and the disc: these are mass-produced and the tolerances cannot be made unacceptably small.

We shall consider one example here to illustrate the way in which such tolerances affect the design; specifically, the choice of the spot diameter $d$ is discussed. We define $d$ as the half-value diameter for the light intensity; we have

$$d = 0.5\lambda/NA,$$

where $\lambda$ is the wavelength of the laser light and $NA$ is the numerical aperture of the objective. To achieve a high information density (see (1.1)) $d$ must be as small as possible. The laser chosen for this system is a small AlGaAs laser, which is inexpensive and only requires a low voltage; the wavelength is thus fixed; $\lambda \approx 800$ nm. This means that in order to have a small spot diameter, we must make the numerical aperture as large as possible. With increasing $NA$, however, the manufacturing tolerances of the player and the disc rapidly become smaller. For example, the tolerance in the local skew of the disc (the disc tilt) relative to the objective-lens axis is proportional to $NA^{-3}$. The tolerance for the disc thickness is proportional to $NA^{-4}$, and the depth of focus, which governs the focusing tolerance, is proportional to $NA^{-2}$. After considering all these factors in relation to one another, we arrived at a value of 0.45 for $NA$. We thus find a value of approximately 1 $\mu$m for the spot diameter $d$.

In the laboratory, the quality of the channel is evaluated by means of an eye pattern, which is obtained by connecting the point Q in Figure 1.1 to an oscilloscope synchronized with the clock for the bit stream $B_0$, see Figure 1.3a. The signals originating from different pits and lands are superimposed on the screen; they are strongly rounded, mainly because the spot diameter is not zero and the pit walls are not vertical. If the transmission quality is adequate, however, it is always possible to determine whether the signal is positive or negative at the lock times (the dashes in Figure 1.3a), and hence to reconstitute the bit stream. The lozenge pattern around a dash in this case is called the eye. The eye pattern indicates the operating margins; the horizontal opening is the time margin and the vertical opening is the amplitude margin. Owing to channel imperfections the eye can become obscured; owing to phase jitter of the signal relative to the clock an eye becomes narrower, and noise reduces its height. The signals in Figure 1.3a were calculated for a perfect
FIGURE 1.3 - Eye pattern. The figures give the read signal (at Q in Figure 1.1) on an oscilloscope synchronized with the EFM bit clock. At the decision instants (marked by dashes) it must be possible to determine whether the signal is above or below the decision level (DL). The three eye patterns have been calculated for a) an ideal optical system, b) a defocusing of 2 micron and c) a defocusing of 2 micron and a disc tilt of 1.2 degrees. The curves give a good picture of experimental results.

optical system. Figure 1.3b shows the effect of defocusing by 2 μm and Figure 1.3c shows the effect of a radial tilt of 1.2° in addition to the defocusing. In Figure 1.3b a correct decision is still possible, but not in Figure 1.3c.

This example also gives some idea of the exacting requirements that the equipment has to meet. A more general picture can be obtained from Table 1.2, which gives the manufacturing tolerances of a number of
relevant parameters, both for the player and for the disc. The list is far from complete, of course.

<table>
<thead>
<tr>
<th>Player</th>
<th>Objective-lens tilt ± 0.2°</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Tracking ± 0.1 μm</td>
</tr>
<tr>
<td></td>
<td>Focusing ± 0.5 μm</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Disc</th>
<th>Thickness 1.2 ± 0.1 mm</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Flatness ± 0.6°</td>
</tr>
<tr>
<td></td>
<td>Pit-edge positioning ± 50 nm</td>
</tr>
<tr>
<td></td>
<td>Pit depth 120 ± 10 nm</td>
</tr>
</tbody>
</table>

With properly manufactured players and discs the channel quality can still be impaired by dirt and scratches forming on the discs during use. By its nature the system is fairly insensitive to these, and any errors they may introduce can nearly always be corrected or masked. In the following, we shall see that the recording code also helps to reduce the sensitivity of imperfections.

1.3 Channel Coding

The playing time of a disc is equal to the track length divided by the track velocity \( v \). For a given disc size the playing time therefore increases if we decrease the track velocity in the system (the track velocity of the master disc and of the user disc). If we do this, the channel becomes worse: the eye height decreases and the system becomes more sensitive to perturbations. The anticipated level of noise and perturbation sets a lower limit to the eye height and thus sets a lower limit to the track velocity. We shall now show that we can decrease this lower limit by an appropriate recording code.

We first consider the situation without the use of a recording code. The incoming data bit stream is an arbitrary sequence of 'ones' and 'zeros'. We consider a group of eight data bits in which the change of bit value is fastest (Figure 1.4a). Uncoded recording (1: pit; 0: land, or vice versa) then gives the pattern of Figure 1.4b. This results in the rounded-off signal of Figure 1.4c at Q in Figure 1.1. Figure 1.4d gives the eye pattern. The signal in Figure 1.4c represents the highest frequency, denoted by \( f_{\text{m}} \), for this mode of transmission, and we have
FIGURE 1.4 - Direct recording of the data bit stream on the disc, no channel coding used. a) Data bit stream of the highest frequency that can occur. b) Direct translation of the bit stream into a pattern of pits. c) The corresponding output signal (at Q in Figure 1.1). d) The eye pattern that follows from (c).

\[ f_{m1} = \frac{f_d}{2}, \]  where \( f_d \) is the data bit rate. The half eye height \( a_1 \) is equal to the amplitude \( A_1 \) of the highest-frequency signal.

The relation between the eye height and the track velocity now follows indirectly from the amplitude-frequency characteristic of the channel; see Figure 1.5. In this diagram, \( A \) is the amplitude of the sinusoidal signal at Q in Figure 1.1 when a sinusoidal unit signal of frequency \( f \) is presented at P. With the aid of Fourier analysis and synthesis the output signal can be calculated from \( A(f) \) for any input signal. The line in the diagram represents a channel with a perfect optical system. In the first part of this section we shall take this for granted. The true situation will always be less favourable. The cut-off frequency is governed by the spot diameter and the track velocity \( v \), in the ideal case:

\[ f_c = (2NA/\lambda)v = v/d. \]
FIGURE 1.5 - Amplitude characteristic of the channel. The diagram gives the amplitude $A$ of the sinusoidal signal at $Q$ (Figure 1.1) when a sinusoidal unit signal is presented at $P$ as a function of the frequency $f$.

For a given track velocity we now obtain the half eye height $a_1$ in Figure 1.4 directly from Figure 1.5: it is equal to the amplitude $A_1$ at the frequency $f_{m1}$. If $v$, and hence $f_c$, is varied, the line in Figure 1.5 rotates about the point 1 on the $A$-axis. For a given minimum value of $a_1$, the figure indicates how far $f_c$ can be decreased; this establishes the lower limit for the track velocity $v$. In particular, if the minimum value for $a_1$ is very small, $f_c$ can be decreased to a value slightly above $f_{m1} = f_d/2$.

Figure 1.6 depicts the situation with coding: an imaginary $8 \to 16$ code, which is very close to EFM, however. Each group of eight incoming data bits (Figure 1.6a) is converted into sixteen channel bits (Figure 1.6a'). This is done by using a codebook that allots unambiguously but otherwise arbitrarily to each word of eight bits a word of sixteen bits, but in such a way that the resultant channel bit stream only produces pits and lands that are at least three channel bits long (Figure 1.6). On the time scale the minimum pit and land lengths (the minimum runlength $T_{min}$) have become $3/2$ times as long as in Figure 1.4, but a simple calculation shows that about as much information can nevertheless be transmitted as in Figure 1.4 (256 combinations for 8 data bits), because there is a greater choice of pit-edge positions per unit length (see Figures 1.6b and 1.6b'); the channel bit length $T_c$ has decreased by a half.

With the recording code we have managed to reduce the highest frequency $f_{m2}$ in the signal (see Figure 1.6c, left; $f_{m2} = f_d/3 = 2f_{m1}/3$). Therefore, $f_c$ and $v$ can be reduced by a factor of $3/2$ for the case in which
FIGURE 1.6 - Eight-to-sixteen modulation. Each group of eight data bits (a) is translated with the aid of a dictionary into sixteen channel bits (a'), in such a way that the runlength is equal to at least three channel bits. b) Pattern of bits produced from the bit stream (a'). b') Pattern of pits obtained with a different input signal. c) The read signal corresponding to (b). d) The resultant eye pattern.

a very small eye height is tolerable (see Figure 1.5); this represents an increase of 50% in playing time.

The recording code also has its disadvantages. In the first place the half eye height ($a_2$) in this case is only half of the amplitude ($A_2$) of the signal at the highest frequency (see Figure 1.6d). This has consequences if the minimum allowable eye height is not very small. For example, the application of the recording code becomes completely unusable if the half eye height in Figure 1.5 has to remain larger than $1/2$ ($a_2 > 1/2$ implies $A_2 > 1$), uncoded recording is then still possible ($A_1 = a_1$). In the second place, the tolerance for time errors and for the positioning of pit edges, together with the eye width, denoted by $T_e$, has decreased by a half. In the process of devising a system, the various factors have to be carefully weighed against one another.
FIGURE 1.7 - Half eye height $a$ as a function of the linear information density $\sigma$, for $8 \rightarrow 8$, $8 \rightarrow 16$ and $8 \rightarrow 24$ code.

To show qualitatively how a choice can be made, we have plotted the half eye height in Figure 1.7 as a function of the linear information density $\sigma$ (the number of incoming data bits per unit length of the track; $\sigma = f_d/v$) for three systems: '8 → 8 code', i.e. uncoded recording, 8 → 16 code, and a system that also has about the same information capacity (256 combinations for 8 data bits) in which, however, the minimum runlength has been increased still further, again at the expense of eye width of course (8 → 24 code, $T_{\min} = 2T$, $T_c = T/3$). The figure is a direct consequence of the reasoning above, with the assumption that the cut-off frequency is 20% lower than the ideal value $(2NA/\lambda)v$, as a first rough adjustment to what we find in practice for the function $A(f)$. From Figure 1.7, another important reason emerges for using the 8 → 16 system. We notice that the slope of the curve belonging to the 8 → 16 code is smaller than that for the uncoded situation. In other words, the coded system is less susceptible to variations of the information density than the uncoded system. This a very important virtue of the coded system, since the system becomes more robust against defocusing and disc skewing, which increase the size of the read-out spot, and thus increase the relative information density.

In qualitative terms, the 8 → 16 system has been chosen because the nature of the noise and other perturbations is such that the eye can be smaller than at A in Figure 1.7 but becomes too small at C. An
improvement is therefore possible with the $8 \to 16$ code, but not with the $8 \to 24$ code.

For our Compact Disc system we have $\sigma = 1.55$ data bits/\(\mu\)m \((f_s = 1.94 \text{ Mb/s}, v = 1.25 \text{ m/s})\), the operating point would therefore be at P in Figure 1.7. The model used is however rather crude and in better models A, B, and C lie more to the left, so that P approaches C. But the $8 \to 16$ code is still preferable to the $8 \to 24$ code, even close to C, since the eye width is 3/2 times as large as for the $8 \to 24$ code.

EFM is a refinement of the $8 \to 16$ code. It has been chosen on the basis of more detailed models and many experiments. At the eye height used, it gives a gain of 25% in information density, compared with uncoded recording.

\section*{1.4 Further Requirements for the Recording Code}

In developing the recording code further, we still had two more requirements to take into account.

In the first place, it must be possible to regenerate the bit clock in the player from the read-out signal (the signal at Q in Figure 1.1). To permit this the number of pit edges per second must be sufficiently large, and in particular the maximum runlength $T_{\text{max}}$ must be as small as possible.

![Image](image.png)

\textbf{FIGURE 1.8} - Read-out signal for six pit edges on the disc, a) for a clean disc, b) for a soiled disc, c) for a soiled disc after the low frequencies have been filtered out. DL is decision level. Because of the soiling, both amplitude and the signal level decrease; the decision errors that this would cause are eliminated by the filter.

The second requirement relates to the 'low-frequency content' of the read signal. This has to be as small as possible. There are two reasons for this. In the first place, the servo systems for track following and focusing are controlled by low-frequency signals, so that low-frequency components of the information signal could interfere with the servo-systems. The second reason is illustrated in Figure 1.8, in which the
read signal is shown for a clean disc (a) and for a disc that has been soiled, e.g., by fingermarks (b). This causes the amplitude and average level of the signal to fall. The fall in level causes a completely wrong read-out if the signal falls below the decision level. Errors of this type are avoided by eliminating the low-frequency components with a filter (c), but the use of such a filter is only permissible provided the information signal itself contains no low-frequency components. In the Compact Disc system the frequency range from 20 kHz to 1.5 MHz is used for information transmission; the servo systems operate on signals in the range 0-20 kHz.

1.5 EFM Recording Code

Figure 1.9 gives a schematic general picture of the bit streams in the encoding system. The information is divided into frames. One frame contains 6 sampling periods, each of 32 audio bits (16 bits for each of the two audio channels). These 32 audio bits are divided into four symbols of eight bits. The bit stream $B_1$ thus contains 24 symbols per frame. In $B_2$, eight parity symbols and one C&D symbol have been added to each frame, resulting in 33 data symbols. The modulator translates each symbol into a new symbol of 14 bits. Added to these are three merging bits, for reasons that will appear shortly. After the addition of a synchronization symbol of 27 bits to the frame, the bit stream $B_i$ is obtained. $B_i$ therefore contains $33 \times 17 + 27 = 588$ channel bits per frame. Finally, $B_i$ is converted into a control signal for the write laser. It should be noted that in $B_i$ '1' or '0' does not mean 'pit' or 'land', as we assumed for simplicity in Figure 1.6, but a '1' indicates a pit edge. The information is thus completely recorded by the position of the pit edges; it therefore makes no difference to the decoding system if 'pit' and 'land' are interchanged on the disc.

Opting for the translation of series of eight bits following the division into symbols in the parity coding has the effect of avoiding error propagation. This is because in the error-correction system an entire symbol is always either 'wrong' or 'not wrong'. One channel-bit error that occurs in the transmission spoils an entire symbol, but - because of the correspondence between channel symbols and data symbols - never more than one symbol. If a different recording code is used, in which the data bits are not translated in groups of eight, but in groups of six or ten, say, then the bit stream $B_2$ is in fact first divided up into six or ten bit
channel symbols. Although one channel-bit error then spoils only one channel symbol, it usually spoils two of the original 8-bit symbols.

In EFM, the data bits are translated eight at a time into 14 channel bits, with a $T_{\text{min}}$ of 3 and a $T_{\text{max}}$ of 11 channel bits (this means at least 2 and at the most 10 successive ‘zeros’ in $B_i$). This choice came about more or less as follows. We have already seen that the choice of about $3/2$ data bits for $T_{\text{min}}$, with about sixteen channel bits on eight data bits, is about the optimum for the Compact Disc system. A simple calculation shows
that at least 14 channel bits are necessary for the reproduction of all the 256 possible symbols of 8 data bits under the conditions $T_{\text{min}} = 3$, $T_{\text{max}} = 11$ channel bits. The choice of $T_{\text{max}}$ was dictated by the fact that a larger choice does not make things very much easier, whereas a smaller choice does create far more difficulties.

With 14 channel bits it is possible to make up 267 symbols that satisfy the runlength conditions. Since we only require 256, we omitted 10 that would have introduced difficulties with the merging of symbols under these conditions, and one other chosen at random. In order to reduce the complexity of the decoder logic, the relationship between data patterns and code patterns have to be optimized. The codebook was compiled with the aid of computer logic optimization in such a way that the translation in the player can be carried out with the simplest possible circuit, i.e., a circuit that contains the minimum of logic gates. In the CD player the EFM conversion can be performed with a programmed logic array of approximately 52 logic functions. Part of the EFM coding table is presented in Table 1.3, which shows the decimal representation of the 8-bit source word (left column) and its 14-bit channel representation (right column).

**TABLE 1.3 - Part of the EFM Encoding Table**

<table>
<thead>
<tr>
<th>Data</th>
<th>Code</th>
<th>Data</th>
<th>Code</th>
</tr>
</thead>
<tbody>
<tr>
<td>100</td>
<td>010001001000010</td>
<td>114</td>
<td>100100100000010</td>
</tr>
<tr>
<td>101</td>
<td>00000000100010</td>
<td>115</td>
<td>01000001000010</td>
</tr>
<tr>
<td>102</td>
<td>01000001000010</td>
<td>116</td>
<td>01000001000010</td>
</tr>
<tr>
<td>103</td>
<td>00100100100010</td>
<td>117</td>
<td>000000100000010</td>
</tr>
<tr>
<td>104</td>
<td>01001001000010</td>
<td>118</td>
<td>00100001000010</td>
</tr>
<tr>
<td>105</td>
<td>10000001000010</td>
<td>119</td>
<td>00100001000010</td>
</tr>
<tr>
<td>106</td>
<td>100100001000010</td>
<td>120</td>
<td>01001000000010</td>
</tr>
<tr>
<td>107</td>
<td>10001001000010</td>
<td>121</td>
<td>00010010010000</td>
</tr>
<tr>
<td>108</td>
<td>010000100000010</td>
<td>122</td>
<td>10010000000010</td>
</tr>
<tr>
<td>109</td>
<td>00000010000010</td>
<td>123</td>
<td>10001000000010</td>
</tr>
<tr>
<td>110</td>
<td>00100010000010</td>
<td>124</td>
<td>01000000000010</td>
</tr>
<tr>
<td>111</td>
<td>001000010000010</td>
<td>125</td>
<td>00010000000010</td>
</tr>
<tr>
<td>112</td>
<td>10000000100010</td>
<td>126</td>
<td>00100001000010</td>
</tr>
<tr>
<td>113</td>
<td>100000100000010</td>
<td>127</td>
<td>01000000000010</td>
</tr>
</tbody>
</table>

The merging bits are primarily intended to ensure that the runlength conditions continue to be satisfied when the symbols are merged. If the runlength is in danger of becoming too short, we choose 'zeros' for the
merging bits; if it is too long we choose a 'one' for one of them. If we do this, we still retain a large measure of freedom in the choice of the merging bits, and we use this freedom to minimize the low-frequency content of the signal. In itself, (see for more details the treatment of runlength limited sequences provided in Chapter 3) two merging bits would be sufficient for continuing to satisfy the runlength conditions. A third merging bit is necessary, however, to give sufficient freedom for effective suppression of low-frequency content, even though it entails a loss of six percent of the information density on the disc. The merging bits contain no audio information, and they are removed from the bit stream in the demodulator.

**FIGURE 1.10** - Strategy for minimizing the running digital sum (RDS). The encoder chooses the merging bits that gives the lowest absolute value of the RDS at the end of a new codeword.

Figure 1.10 illustrates, finally, how the merging bits are determined. Our measure of the low-frequency content is the *running digital sum* (RDS); this is the difference between the totals of pit and land lengths accumulated from the beginning of the disc. At the top are shown two data symbols of \( B_2 \) and their translation from the codebook into channel symbols \( B_3 \). From the \( T_{\text{min}} \) rule the first of the merging bits in this case must be a 'zero'; this position is marked 'x'. In the two following positions the choice is free; these are marked 'm'. The three possible
choices $x_{mm} = 000$, 010, and 001 would give rise to the patterns of pits as illustrated, and to the indicated waveform of the RDS, on the assumption that the RDS was equal to 0 at the beginning. The system now opts for the merging combination that makes the RDS at the end of the second symbol as close to zero as possible, i.e., 000 in this case. If the initial value had been -3, the merging combination 001 would have been chosen.

When this strategy is applied, the noise in the servo-band frequencies ($< 20$ kHz) is suppressed by about 10 dB. In principle better results can be obtained, within the agreed standard for the Compact Disc system, by looking more than one symbol ahead, since minimization of the RDS in the short term does not always contribute to longer-term minimization.
Many structures encountered in communication channels with input constraints can usefully be modelled in terms of a Markov process. A knowledge of the properties of sequences emitted by a Markov information source is essential to understand the basic trade-offs between the prescribed input constraints and the concomitant properties of a code. The aim of this introductory but basic chapter is to provide a review of those aspects of discrete-time Markov chain theory most commonly encountered in spectral coding applications.

2.1 Introduction

Our objective in this chapter is to provide an answer to a fundamental question that arises in the analysis of communication systems: given an information source, how do we quantify the amount of information that the source is emitting? Information sources can be classified in two categories: continuous and discrete. Continuous information sources emit a continuous-amplitude, continuous-time waveform, whereas a discrete information source conveys symbols from a finite set of letters or alphabet of symbols. In this text we shall confine ourselves to discrete information sources. Every message emitted by the source contains some information, but some messages convey more information than others. In order to quantify the information content of messages transmitted by information sources, we will study the important notion of a measure of information in this chapter. The material presented in Section 2.2 is based on the pioneering work of Shannon, who published in 1948 a treatise on the mathematical theory of communications.

The physical limitations of, for instance, the time-base regeneration or the adaptive equalization lead to the simple conclusion that sequences that may foil the receiver circuitry, have to be discarded, and that therefore not the entire population of possible sequences can be used. To
be specific, sequences are allowed that guarantee runs of 'zero' symbols that are neither too short nor too long. This is an example of a channel with runlength constraints (a runlength is the number of times that the same symbol is repeated). A channel that does not permit input sequences containing any of a certain collection of forbidden subsequences is called an input-restricted or constrained channel. It should be appreciated that the channel constraints considered here are deterministic by nature; the basic assumption is that the messages are conveyed over a noiseless channel, that is, a channel that never makes errors. Our concern here is to maximize the number of messages that can be sent (or stored) over the channel in a given time (or given area) given the deterministic channel constraints. On the average, \( m \) binary user symbols are translated into \( n \) binary channel symbols. Obviously, since not the entire population is used, \( n \geq m \). A measure of the efficiency implied by a particular code is given by the code rate \( R \), where \( R \) is defined as \( R = \frac{m}{n} \). The fraction of transmitted symbols that are redundant is \( 1 - R \). Clearly, an unconstrained channel, that is a channel that permits any arbitrary binary sequence, has unity rate. It is a cardinal question how many of the possible sequences have to be excluded, or, stated alternatively, how much of a rate (loss) one needs to suffer to convert arbitrary binary data into a sequence that satisfies the specified channel constraints. The maximum rate feasible given determinate constraints is often called, in honour of the founder of information theory, the Shannon capacity of the noiseless input-constrained channel. The central concept of channel capacity is introduced in Section 2.3. The references quoted may be consulted for more details of the present topic. We commence with the general theory.

2.2 Information Content of a Message, Entropy

Messages produced by discrete information sources consist of sequences of symbols. The simplest model of an information source is probably a source that emits symbols in a statistically independent sequence, with the probabilities of occurrence of various symbols being invariant with respect to time.

2.2.1 Entropy of Memoryless Sources

This section applies to sources that emit symbols in statistically independent sequences. Such a source is usually called memoryless. We
assume that the symbol transmitted by the source is the result of a probabilistic experiment. Suppose that the probabilistic experiment involves the observation of a discrete random variable denoted by \( X \). Let \( X \) take on one of a countable set of possible values \( \{ x_1, x_2, \ldots, x_M \} \) with probabilities \( p_1, p_2, \ldots, p_M \), respectively. Clearly,

\[
\sum_{i=1}^{M} p_i = 1.
\]

The outcomes of the experiments are emitted by the source, which generates a sequence denoted by \( \{ X \} = \{ \ldots, X_{-1}, X_0, X_1, \ldots \} \). We may imagine the symbols being produced at a uniform rate, one per unit of time, with \( X_t \) produced at time \( t \).

We now consider a formulation of the information content in terms of information theory. It is not possible here to consider in detail the formal establishment of a measure of information. Again, the references provided at the end of this chapter may be consulted for more details of the present topic. The fundamental notion in information theory is that of surprise or uncertainty; unlikely events carrying more information than likely ones, and vice versa. A measure of information content of a random variable, usually called uncertainty or entropy, was proposed by Shannon [1]:

\[
H(p_1, \ldots, p_M) = -\sum_{i=1}^{M} p_i \log_2 p_i, \quad 0 \leq p_i \leq 1.
\] (2.1)

The base of the logarithm in (2.1) is arbitrary but will, of course, affect the numerical value of the entropy \( H(p_1, \ldots, p_M) \). When the logarithm is taken to the base \( e \), the information is measured in units of nats (a short-hand notation of natural units). The 2-base is in common use. If we employ base two, the units of \( H(p_1, \ldots, p_M) \) are expressed in bits, a convenient and easily appreciated unit of information. The conversion factor is one nat equals 1.443 bits.

If \( M = 2 \), one usually writes \( h(p) \) instead of \( H(p, 1 - p) \). It follows that

\[
h(p) = -p \log_2 p - (1 - p) \log_2 (1 - p), \quad 0 \leq p \leq 1,
\] (2.2)

where by definition \( h(0) = h(1) = 0 \) to make \( h(p) \) continuous in the closed interval \([0,1]\). The function \( h(p) \) is often called the binary entropy function. Figure 2.1 shows a sketch of \( h(p) \). We notice that the
The information content of a stochastic variable reaches a maximum when \( p = 1/2 \).

**Example 2.1:** Consider the flipping of a coin. Let \( \Pr(\text{heads}) = p \) and \( \Pr(\text{tails}) = 1 - p \), \( 0 \leq p \leq 1 \), then the entropy is given by (2.2). That \( h(1/2) = 1 \) is of course intuitively confirmed by the fact that one requires one bit to represent the outcome of the tossing of a fair coin. For example, the outcome *heads* is represented by a logical 'zero' and *tails* by a logical 'one'. Since \( h(1/4) \approx 0.811 \), one expects that on the average only 811 bits are needed to represent the outcomes of 1000 tossings with this 'unfair' coin.

How this is done in an efficient fashion is the domain of source coding, and, although an important subject in its own right, is not further pursued in this text (see for example [2]).

### 2.2.2 Markov Chains

In the previous section we introduced the notion of entropy in a simple situation, where it was assumed that the symbols are independent and occur with fixed probabilities. That is, the occurrence of a specific symbol
at a certain instant does not alter the probability of occurrences of symbols during any other symbol intervals. We need to extend the concept of entropy for more complicated structures where symbols are not chosen independently but their probabilities of occurring depend on preceding symbols. It is to be emphasized that nearly all practical sources emit sequences of symbols that are statistically dependent. Sequences formed by the English language are an excellent example. Occurrence of letter Q implies that the letter to follow is probably a U. Regardless of the form of the statistical dependence, or structure, among the successive source outputs, the effect is that the amount of information coming out of such a source is smaller than from a source emitting the same set of characters in independent sequences. The development of a model for sources with memory is the focus of the ensuing discussion.

In probability theory the notation

\[ Pr(A | B) \]

means the probability of occurrence of event \( A \) given that (\( | \)) event \( B \) has occurred. Many of the structures that will be encountered in the subsequent chapters can usefully be modelled in terms of a *Markov chain*. A Markov chain is a special type of stochastic process distinguished by a certain Markov property. A (discrete) Markov chain is defined as a discrete random process of the form

\[
\{ ... , Z_{-2}, Z_{-1}, Z_0, Z_1, ... \},
\]

where the variables \( Z_t \) are dependent discrete random variables taking values in the state alphabet \( \Sigma = \{ \sigma_1, ..., \sigma_N \} \), and the dependence satisfies the *Markov condition*

\[
Pr(Z_t = \sigma_j | Z_{t-1}, Z_{t-2}, ...) = Pr(Z_t = \sigma_j | Z_{t-1}). \tag{2.3}
\]

In words, the variable \( Z_t \) is independent of past samples \( Z_{t-2}, Z_{t-3}, ... \) if the value of \( Z_{t-1} \) is known. A Markov chain can be described by a *transition probability matrix* \( Q \) with elements

\[
q_{ij} = Pr(Z_t = \sigma_j | Z_{t-1} = \sigma_i), \quad 1 \leq ij \leq N. \tag{2.4}
\]

The transition probability matrix \( Q \) is a *stochastic matrix*, that is, (i) its entries are non-negative, and (ii) the entries of each row sum to one. Any stochastic matrix constitutes a valid transition probability matrix.

Imagine the process starts at time \( t = 1 \) by choosing an initial state in accordance with a specified probability distribution. If we are in state
If \( \sigma_i \) at time \( t = 1 \), then the process moves at \( t = 2 \) to a possibly new state, the quantity \( q_{ij} \) is the probability that the process will move to state \( \sigma_j \) at time \( t = 2 \). If we are in state \( \sigma_j \) at instant \( t = 2 \), we move to \( \sigma_k \) at instant \( t = 3 \) with probability \( q_{jk} \). This procedure is repeated ad infinitum.

The state-transition diagram of a Markov chain, which is portrayed in Figure 2.2a, represents a Markov chain as a graph where the states are embodied by the nodes of the graph, the transition between states is represented by a directed line from the initial to the final state. The transition probabilities \( q_{ij} \) corresponding to various transitions are shown marked along the lines of the graph. Another useful representation of a Markov chain is provided by a *trellis (or lattice) diagram* (see Figure 2.2b). This is a state diagram augmented by a time axis so that it provides for easy visualization of how the states change with time.

![Figure 2.2](image)

**FIGURE 2.2** - Alternative representations of a three-state Markov chain. (a) state-transition diagram, (b) trellis diagram for the same chain.

In theory, there are many types of Markov chains; here we restrict our attention to chains that are *ergodic* and *regular* [3]. Roughly speaking, ergodicity means that from any state the chain can eventually reach any other state; regularity means that the Markov chain is non-periodic. In the practical structures that will be encountered, all these conditions hold.

We shall now take a closer look at the dynamics of a Markov chain. To that end, let \( w^0_j = \Pr(Z_t = \sigma_j) \) represent the probability of being in state \( \sigma_j \) at time \( t \). Clearly,
The probability of being in state \( \sigma_j \) at time \( t \) may be expressed in the state probabilities at instant \( t-1 \):

\[
w_j^{(t)} = w_1^{(t-1)}q_{1j} + w_2^{(t-1)}q_{2j} + \cdots + w_N^{(t-1)}q_{Nj}.
\]  

(2.6)

The previous equation suggests the use of matrices. If we introduce the state distribution vector

\[
w^{(t)} = (w_1^{(t)}, \ldots, w_N^{(t)}),
\]

then the previous equation may conveniently be expressed in an elegant matrix/vector notation, thus

\[
w^{(t)} = w^{(t-1)}Q.
\]  

(2.7)

By iteration we obtain

\[
w^{(t)} = w^{(1)}Q^{t-1}.
\]  

(2.8)

In other words, the state distribution vector at time \( t \) is the product of the state distribution vector at time \( t = 1 \), and the \((t - 1)\)th power of the transition matrix. It is easy to see that \( Q^{t-1} \) is also a stochastic matrix. Formula (2.8) is equivalent to the assertion that the \( n \)-step transition matrix is the \( n \)th power of the single step transition matrix \( Q \). We note also that \( Q^0 = I \) is the ordinary identity matrix.

We shall concentrate now on the limiting behaviour of the state distribution vector as \( t \to \infty \). In many cases of practical interest there is only one such limiting distribution vector, denoted by \( \bar{\pi} = (\pi_1, \ldots, \pi_N) \). In the long run the state distribution vector converges to this equilibrium distribution vector from any valid initial state probability vector \( w^{(1)} \), so

\[
\bar{\pi} = \lim_{t \to \infty} w^{(1)}Q^{t-1}.
\]  

(2.9)

The number \( \pi_i \) is called the steady, or stationary, state probability of state \( \sigma_i \). The equilibrium distribution vector can be obtained by solving the system of linear equations in the \( N \) unknowns \( \pi_1, \ldots, \pi_N \):

\[
\pi Q = \bar{\pi}.
\]  

(2.10)
Only \( N - 1 \) of these \( N \) equations are independent, so we solve the top \( N - 1 \) along with the normalizing condition

\[
\sum_{i=1}^{N} \pi_i = 1.
\]

The proof is elementary: we note that if \( \pi Q = \pi \), then

\[
\pi Q' = \pi QQ'^{-1} = \pi Q'^{-1} = \cdots = \pi.
\]

Decomposition of the initial state vector \( w^{(1)} \) in terms of the eigenvectors of \( Q \) can be convenient to demonstrate the process of convergence. The matrix \( Q \) has \( N \) eigenvalues \( \{ \lambda_1, \ldots, \lambda_N \} \), that can be found by solving the characteristic equation

\[
\det[Q - \lambda I] = 0, \tag{2.11}
\]

where \( I \) is the identity matrix, and \( N \) (left) eigenvectors \( \{ u_1, \ldots, u_N \} \) each of which is a solution of the system

\[
u_i Q = \lambda_i u_i, \quad i = 1, \ldots, N. \tag{2.12}
\]

Provided that \( \lambda_i, i = 1, \ldots, N, \) are distinct, there are \( N \) independent eigenvectors, and the eigenvectors \( u_i, i = 1, \ldots, N, \) constitute a basis. The initial state vector may be written as

\[
w^{(1)} = \sum_{i=1}^{N} a_i u_i. \tag{2.13}
\]

With (2.8) we find the state distribution vector \( w^{(0)} \) at instant \( t \):

\[
w^{(t)} = w^{(1)} Q^{(t-1)} = \sum_{i=1}^{N} a_i \lambda_i^{(t-1)} u_i. \tag{2.14}
\]

If it is assumed that the eigenvalues are distinct, the \( \{ \lambda_i \} \) can be ordered such that \( |\lambda_1| > |\lambda_2| > |\lambda_3| \) etc. Combination of (2.10) and (2.12) reveals that \( \pi \) is an eigenvector with unity eigenvalue, thus \( \lambda_i = 1 \). We then have

\[
w^{(t)} = \pi + \sum_{i=2}^{N} a_i \lambda_i^{(t-1)} u_i \tag{2.15}
\]

and convergence to \( \pi \) is assured since \( |\lambda_i| < 1, \ i \neq 1 \).
2.2.3 Entropy of Markov Information Sources

We are now in the position to describe a Markov information source. Given a finite Markov chain \( \{Z_t\} \) and a function \( \zeta \) whose domain is the set of states of the chain and whose range is a finite set \( \Gamma \), the source alphabet, then the sequence \( \{X_t\} \), where \( X_t = \zeta(Z_t) \), is said to be the output of a Markov information source corresponding to the chain \( \{Z_t\} \) and the function \( \zeta \). In general, the number of states can be larger than the cardinality of the source alphabet, which means that one output symbol may correspond to more than one state.

The essential feature of the Markov information source is that it provides for dependence between successive symbols, which introduces redundancy in the message sequence. Each symbol conveys less information than it is capable of conveying since it is to some extent predictable from the preceding symbol. In the foregoing description of an information source we assumed that the symbol emitted is solely a function of the state that is entered. This type of description is usually called a Moore-type Markov source. In a different description, called the Mealy-type Markov source, the symbols emitted are a function of the Markov chain \( X_t = \zeta(Z_t, Z_{t-1}) \). Both descriptions are equivalent. Let a Mealy-type machine be given. By defining a Markov information source with state set composed of triples \( \{o_{-j}, o_{-j}, (o_{-i}, o_{-j})\} \) and label \( \zeta(o_{-i}, o_{-j}) \) on the state \( \{o_{-j}, o_{-j}, (o_{-i}, o_{-j})\} \) we obtain a Moore-type Markov source. The Moore model is referred to as the edge graph of the Mealy model. An example of a Mealy-type information source and its Moore-type equivalent is shown in Figure 2.3.

![Figure 2.3](image-url)
The idea of a Markov source has enabled us to represent certain types of structure in streams of data. We now examine the information content, or entropy, of a sequence emitted by a Markov source. The entropy of a Markov information source is difficult to compute in most cases. For a certain class of Markov information sources, termed unifilar Markov information source, the computation may be greatly simplified. The word unifilar refers to the following property [4].

Let a Markov information source with a set of states $\Sigma = \{\sigma_1, \ldots, \sigma_N\}$, output alphabet $\Gamma$, and associated output function $\zeta(Z)$ be given. For each state $\sigma_k \in \Sigma$, let $\sigma_{k1}, \sigma_{k2}, \ldots, \sigma_{kn}$ be the states that can be reached in one step from $\sigma_k$, that is, the states $\sigma_j$ such that $q_{kj} > 0$. The source is said to be unifilar if for each state $\sigma_k$ the symbols $\zeta(\sigma_{k1}), \ldots, \zeta(\sigma_{kn})$ are distinct. In other words, each state reachable directly from state $\sigma_k$ is associated with a distinct symbol. Provided this condition is met and the initial state of the Markov information source is known, the sequence of emitted symbols determines the sequence of states followed by the chain and a simple formula is available for the entropy of the emitted $X$-process. Given a unifilar Markov source, as above, let $\sigma_{k1}, \ldots, \sigma_{kn}$ be the states that can be reached in one step from $\sigma_k$, then it is quite natural to define the uncertainty of state $\sigma_k$ as $H_k = H(p_{k1}, \ldots, p_{kn})$, with $H(p_{k1}, \ldots, p_{kn})$ as defined in (2.1). Shannon [1] defined the entropy of the unifilar Markov source as the average of these $H_k$ weighed in accordance with the steady state probability of being in a state in question, that is, by the expression

$$H(X) = \sum_{k=1}^{N} \pi_k H_k.$$  \hfill (2.16)

Note that we use the notation $H(X)$ to express the fact that we are considering the entropy of sequences $\{X\}$ and not the function $H(.)$. The next numerical example may serve to illustrate the theory.

**Example 2.2:** Consider a 3-state unifilar Markov chain with transition probability matrix

$$Q = \begin{bmatrix} 0 & 0 & 1 \\ 1/2 & 1/3 & 1/6 \\ 1/2 & 1/2 & 0 \end{bmatrix}.$$ 

What is the average probability of being in one of the three states? We find, using (2.10), the following system of linear equations that govern the steady state probabilities:
\[
\frac{1}{2} \pi_2 + \frac{1}{2} \pi_3 = \pi_1
\]
\[
\frac{1}{3} \pi_2 + \frac{1}{2} \pi_3 = \pi_2
\]
\[
\pi_1 + \frac{1}{6} \pi_2 = \pi_3.
\]
from which we obtain \( \pi_3 = \frac{4}{3} \pi_2 \) and \( \pi_1 = \frac{7}{6} \pi_2 \). Since \( \pi_1 + \pi_2 + \pi_3 = 1 \) we have
\[
\pi_1 = \frac{1}{3}, \quad \pi_2 = \frac{2}{7}, \quad \pi_3 = \frac{8}{21}.
\]
The entropy of the information source depicted in Figure 2.2 is found with (2.1) and (2.16)
\[
H\{X\} = \frac{1}{3} H(1) + \frac{2}{7} H\left(\frac{1}{2}, \frac{1}{3}, \frac{1}{6}\right) + \frac{8}{21} H\left(\frac{1}{2}, \frac{1}{2}\right) \approx 0.798.
\]

2.3 Capacity of Discrete Noiseless Channels

In this section we compute the maximum information rate, capacity, that can be sent over the constrained channel in a given time. Shannon [1] defined the capacity \( C \) of a discrete noiseless channel by
\[
C = \lim_{T \to \infty} \frac{\log_2 N(T)}{T},
\]
where \( N(T) \) is the number of permitted signals of duration \( T \). The problem of calculation of the capacity for constrained channels is in essence a combinatorial problem, that is, finding the number of allowed sequences \( N(T) \). This fundamental definition will be worked out in a moment for some specific channel models. We start, since virtually all channel constraints can be modelled as such, with the computation of the capacity of Markov information sources.

2.3.1 Capacity of Markov Information Sources

In the previous sections we developed a measure of information content of an information source that can be represented by a finite Markov model. As discussed, the measure of information content, entropy, can
be expressed in terms of the limiting state-transition probabilities and the conditional entropy of the states. In this section we address a problem that provides the key to many questions that will emerge in the chapters to follow. Given a unifilar $N$-state Markov source with states $\{\sigma_1, \ldots, \sigma_N\}$ and transition probabilities $\hat{q}_{ij}$ we define the connection matrix $D = \{d_{ij}\}$ of the source as follows. Let

$$d_{ij} = 1 \text{ if } \hat{q}_{ij} > 0$$
$$d_{ij} = 0 \text{ if } \hat{q}_{ij} = 0, \quad i, j = 1, \ldots, N.$$ 

In other words, the connection (or adjacency) matrix contains binary-valued elements, and it is formed by replacing the positive elements of the transition matrix by 1's. For an $N$-state source, the connection matrix $D$ is defined by $d_{ij} = 1$ if a transition from state $i$ to state $j$ is allowable and $d_{ij} = 0$ otherwise. For a given connection matrix, we wish to choose the transition probabilities in such a way that the entropy

$$H(X) = \sum_{k=1}^{N} \pi_k H_k$$

is maximized. The maximum entropy of a unifilar Markov information source, given its connection matrix, is given by

$$C = \max H(X) = \log_2 \lambda_{\text{max}}, \quad (2.18)$$

where $\lambda_{\text{max}}$ is the largest eigenvalue of the connection matrix $D$. The existence of a positive eigenvalue and corresponding eigenvector with positive elements is guaranteed by the Perron-Frobenius theorems [5]. Essentially, there are two approaches to prove the preceding equation. One approach, provided by Shannon [1], is a straightforward routine, using Lagrange multipliers, of finding the extreme value of a function of several independent variables. The second proof of equation (2.18), to be followed here, is established by enumerating the number of distinct sequences that a Markov source can generate.

The number of distinct sequences of length $m + 1$ emanating from state $\sigma_i$ is

$$N(m + 1) = \sum_{j=1}^{N} d_{ij} N(m), \quad i = 1, \ldots, N. \quad (2.19)$$
This is a system of linear homogeneous difference equations with constant coefficients, and therefore the solution is a linear combination of exponentials $\lambda^m$. To find the particular $\{\lambda_i\}$ we assume a solution of the form $N(m) = y_i \lambda^m$ to obtain

$$\lambda^m(y_i) = \lambda^m \sum_{j=1}^{N} d_j y_j, \quad i = 1, \ldots, N,$$

or, letting $y^T = (y_1, \ldots, y_N)$, we have

$$\lambda y = Dy.$$

Thus the allowable $\{\lambda_i\}$ are the eigenvalues of the matrix $D$. For large sequence length $m$ we may approximate $N(m)$ by

$$N(m) \approx a m \lambda_{\text{max}},$$

where $a$ is a constant independent of $m$ and $\lambda_{\text{max}}$ is the largest real eigenvalue of the matrix $D$, or in other words, $\lambda_{\text{max}}$ is the largest real root of the determinant equation

$$\det[D - zI] = 0. \quad (2.20)$$

This is not to say that $N(m)$ is accurately determined by the exponential term when $m$ is small. We have

$$\frac{1}{m} \log_2 N(m) \approx \frac{1}{m} (\log_2 a + m \log_2 \lambda_{\text{max}}).$$

The maximum entropy of the noiseless channel is given by (2.17), or

$$C = \lim_{m \to \infty} \frac{1}{m} \log_2 N(m) = \log_2 \lambda_{\text{max}}.$$

The transition probabilities $q_{ij}$ associated with the maximum entropy of the source can be found with the following reasoning. Let $p = (p_1, \ldots, p_N)^T$ denote the eigenvector associated with the eigenvalue $\lambda_{\text{max}}$, or

$$Dp = \lambda_{\text{max}} p.$$

The state-transition probabilities that maximize the entropy are

$$q_{ij} = \lambda_{\text{max}}^{-1} d_{ij} \frac{p_j}{p_i}. \quad (2.21)$$
To prove (2.21) is a matter of substitution. According to the Perron-Frobenius theorems [5], the components of the eigenvector \( \mathbf{p} \) are positive, and thus \( q_{ij} \geq 0, 1 \leq i, j \leq N \). Since \( \mathbf{p} = (p_1, \ldots, p_N) \) is an eigenvector for \( \lambda_{\text{max}} \), we conclude

\[
\sum_{j=1}^{N} q_{ij} = 1,
\]

and hence the matrix \( Q \) is indeed stochastic.

The entropy of a Markov information source is, according to definition (2.16),

\[
H(X) = \sum_{k=1}^{N} \pi_k H_k,
\]

where \( H_k \) is the uncertainty of state \( \sigma_k \) and \( (\pi_1, \ldots, \pi_N) \) is the steady state distribution. Thus

\[
H(X) = -\sum_{i,j=1}^{N} \pi_i q_{ij} \log_2 q_{ij} = \sum_{i,j=1}^{N} \pi_i q_{ij} \left( \log_2 \lambda_{\text{max}} + \log_2 p_i - \log_2 p_j \right).
\]

Since

\[
\sum_{i,j} \pi_i q_{ij} \log_2 p_i = \sum_{i} \pi_i \log_2 p_i
\]

and

\[
\sum_{i,j} \pi_i q_{ij} \log_2 p_j = \sum_{j} \log_2 p_j \sum_{i} \pi_i q_{ij} = \sum_{j} \pi_j \log_2 p_j,
\]

we obtain

\[
H(X) = -\sum_{i,j=1}^{N} \pi_i q_{ij} \log_2 q_{ij} = \sum_{i,j} \pi_i q_{ij} \log_2 \lambda_{\text{max}} = \log_2 \lambda_{\text{max}}.
\]

This demonstrates that the transition probabilities given by (2.21) are indeed maximizing the entropy.

**Example 2.2 (Continued):** Consider again the 3-state unifilar Markov chain with transition probability matrix


\[ Q = \begin{bmatrix} 0 & 0 & 1 \\ 1/2 & 1/3 & 1/6 \\ 1/2 & 1/2 & 0 \end{bmatrix} \]

The adjacency matrix \( D \) is

\[ D = \begin{bmatrix} 0 & 0 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 0 \end{bmatrix} \]

The characteristic equation is

\[
\det[D - zI] = -z(z^2 - z - 2) = -z(z - 2)(z + 1) = 0,
\]

from which we conclude that the largest root is \( \lambda_{\text{max}} = 2 \), and the capacity is \( C = \log_2 \lambda = 1 \). The eigenvector associated with the largest eigenvalue is \( p = (1, 3, 2)^T \). The transition probabilities that maximize the entropy of the Markov information source are found with (2.21):

\[ Q = \begin{bmatrix} 0 & 0 & 1 \\ 1/6 & 1/2 & 1/3 \\ 1/4 & 3/4 & 0 \end{bmatrix} \]

\[ \text{2.3.2 Variable-Length Symbols} \]

A basic assumption made in the previous sections is that all transmitted symbols are of unit duration. In this section we study channels whose symbols do not all have the same duration. The sequences considered are composed of elementary symbols, called phrases, selected from a finite set \( \{X_1, \ldots, X_N\} \). Each of the symbols \( X_1, \ldots, X_N \) is assumed to have a certain duration \( t_1, \ldots, t_n \) that are integer multiples of the unit of time.

We consider again an \( N \)-state source. Let \( t_{ij} \) denote the duration of the \( k \)th symbol that emanates from state \( i \) and is allowed to terminate in state \( j \). Define the \( N \times N \) connection matrix \( D(z) \) by

\[
d_{ij}(z) = \sum_l z^{-t_{ij}}, \quad 1 \leq i, j \leq N.
\]

The capacity of the above channel equals the base-2 logarithm of the largest real root of the characteristic equation
\[ \det[D(z) - I] = 0. \quad (2.23) \]

Eq. (2.23) reduces to (2.20) if all symbols are of unity duration.

If the phrases are emitted independently, then (2.22) and (2.23) reduce to the characteristic equation
\[ P(z) = z^{-t_1} + \cdots + z^{-t_n} = 1. \quad (2.24) \]

From this equation it is clear that \( P(z) \) is a monotonic decreasing function:
\[ P(0) = \infty, \quad P(\infty) = 0. \]

Therefore, the equation \( P(z) = 1 \) cannot have more than one positive root. It is instructive at this point to consider a simple case.

**Example 2.3:** Consider a sequence composed of symbols of duration two, three, and four time units, respectively. According to (2.24), the characteristic equation is
\[ z^{-2} + z^{-3} + z^{-4} = 1, \]
or
\[ z^4 - z^2 - z - 1 = 0. \]

After some computation, we find that the base-2 logarithm of the largest real root of this equation is 0.5515, which is the capacity of the channel.

In the next example we assume the same set of phrases, but now we apply the extra condition that the phrases are not emitted independently: for example a phrase of length two is not allowed to follow a phrase of length two. We can write down the following connection matrix
\[ D(z) = \begin{bmatrix} 0 & z^{-2} & z^{-2} \\ z^{-3} & z^{-3} & z^{-3} \\ z^{-4} & z^{-4} & z^{-4} \end{bmatrix}. \]

Using (2.23), we obtain the equation
\[ \det \begin{bmatrix} -1 & z^{-2} & z^{-2} \\ z^{-3} & -1 & z^{-3} \\ z^{-4} & z^{-4} & -1 \end{bmatrix} = 0, \]
or
The capacity of this channel is 0.469.

Many worked examples of constrained channels will be treated at greater length in the subsequent chapters.

2.3.3 Interleaved Sequences

We consider the time-multiplexing operation which forms a composite signal by repetitively interleaving phrases from two signal sets. Let one sequence be composed of phrases of durations \( t_m \in S_o \), and let the second sequence have phrases of durations \( t_j \in S_e \). The emitted, interleaved, sequence is composed of phrases taken alternately from the first, odd, sequence and the second, even, sequence. Reflection on the fact that the interleaved sequence is composed of phrases of duration \( t_i = t_j + t_m, t_m \in S_o, t_j \in S_e \), will reveal that the characteristic equation is

\[
\left( \sum_{j \in S_e} z^{-j} \right) \left( \sum_{m \in S_o} z^{-m} \right) = 1. \tag{2.25}
\]

It is a simple matter to extend the preceding results to a larger number of multiplexed processes, and it is not further pursued here.

2.4 References

Runlength-Limited Sequences

Codes based on runlength-limited sequences are the state of the art cornerstone of current disk recorders whether their nature is magnetic or optical. In this chapter, we shall provide a detailed description of the limiting properties of runlength-limited sequences, and give a comprehensive review of the construction methods, ad hoc as well as systematic, that are available.

3.1 Introduction

By far the most frequently reported coding schemes applied in recording practice have been those constituted by runlength-limited sequences, and it is with this type of scheme that this chapter is almost exclusively concerned. The length of time usually expressed in channel bits between transitions is known as the runlength. Runlength-limited (RLL) sequences are characterized by two parameters, \((d + 1)\) and \((k + 1)\), which stipulate the minimum and maximum runlength, respectively, that may occur in the sequence. The parameter \(d\) controls the highest transition frequency, and thus has a bearing on intersymbol interference when the sequence is conveyed over a bandwidth-limited channel. In the transmission of binary data it is generally desirable that the received signal is self-synchronizing or self-clocking. Timing is commonly recovered with a phase-locked loop which adjusts the phase of the detection instant according to observed transitions of the received waveform. The maximum runlength parameter \(k\) ensures adequate frequency of transitions for synchronization of the read clock. The grounds on which \(d\) and \(k\) values are chosen, in turn, depend on various factors such as the channel response, the desired data rate (or information density), and the jitter and noise characteristics. Some of the design considerations underlying a certain choice are explained in Chapter 1.
Recording codes that are based on runlength-limited sequences have found almost universal application in optical and magnetic disk recording practice. Archetypes are the rate $1/2$, $(d = 2, k = 7)$ code which is applied in the IBM3380 rigid disk drive, and the EFM code (rate $= 8/17$, $d = 2, k = 10$) which is the basis of the Compact Disc (see Chapter 1).

The theory on runlength-limited sequences is best explained by introducing another constrained sequence, which is closely related to an RLL sequence.

**Definition**: A $dk$-limited binary sequence, in short, $(dk)$ sequence, satisfies simultaneously the following two conditions:

1. $d$ constraint - two logical 'ones' are separated by a run of consecutive 'zeros' of length at least $d$.
2. $k$ constraint - any run of consecutive 'zeros' is of length at most $k$.

If only condition 1 is satisfied, the sequence is said to be $d$-limited (with $k = \infty$), and will be termed $(d)$ sequence.

In general, a $(dk)$ sequence is not employed in optical or magnetic recording without a simple precoding step. A $(dk)$ sequence is converted to a runlength-limited channel sequence in the following way. Let the channel signals be represented by a sequence $\{y_t\}, y_t \in \{-1,1\}$. The channel signals represent the positive or negative magnetization of the recording medium, or pits or lands when dealing with optical recording. The logical 'ones' in the $(dk)$ sequence indicate the positions of a transition $1 \rightarrow -1$ or $-1 \rightarrow 1$ of the corresponding runlength-limited sequence. The binary $(dk)$ sequence

$$0 1 0 0 0 1 0 0 1 0 0 0 1 1 0 1 \ldots$$

would be converted to the RLL channel sequence:

$$1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 \ldots$$

The mapping of the waveform by the precoding step is known as the non-return-to-zero-inverse, (NRZI) data encoding method, or *change of state* encoding. Waveforms that are transmitted without such an intermediate coding step are referred to as non-return-to-zero (NRZ). The name stems from telegraphy, and has no meaning in relation to recording channels. Coding techniques using the NRZI format are generally accepted in digital optical and magnetic recording practice. It can readily be verified that the minimum and maximum distance between consecutive transitions of the RLL sequence derived from a
(dk) sequence is \( d + 1 \) and \( k + 1 \) symbols, respectively, or in other words, the RLL sequence has the virtue that at least \( d + 1 \) and at most \( k + 1 \) consecutive like symbols occur.

In Table 3.1 we have collected some parameters of runlength-limited codes that have found practical application. The characteristics of the various codes will be explained later.

<table>
<thead>
<tr>
<th>( d )</th>
<th>( k )</th>
<th>( R )</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>1</td>
<td>1/2</td>
</tr>
<tr>
<td>1</td>
<td>3</td>
<td>1/2</td>
</tr>
<tr>
<td>2</td>
<td>7</td>
<td>1/2</td>
</tr>
<tr>
<td>1</td>
<td>7</td>
<td>2/3</td>
</tr>
<tr>
<td>2</td>
<td>11</td>
<td>1/2</td>
</tr>
<tr>
<td>2</td>
<td>10</td>
<td>8/17</td>
</tr>
</tbody>
</table>

The outline of this chapter is as follows. Section 3.2 addresses the problem of counting the number of binary runlength-limited sequences of a certain, given, length. Of particular note are sections 3.3-5 which deal with the properties of maxentropic runlength-limited sequences. We first compute the asymptotic information rate of RLL sequences and then turn to the description of the statistical characteristics of maxentropic RLL sequences. The remaining sections deal with the operating principles and design methods of practicable schemes for encoding and decoding RLL sequences.

### 3.2 Number of Binary (dk) Sequences Versus Sequence Length \( n \)

In this section we address the problem of counting the number of sequences of a certain length which comply with given \( d \) and \( k \) constraints. Let \( N_d(n) \) denote the number of distinct \( (d) \) sequences of length \( n \), and define

\[
N_d(n) = 0, \quad n < 0, \\
N_d(0) = 1. \quad (3.1)
\]

The number of \( (d) \) sequences of length \( n > 0 \) is found with the recursive relations
\[ N_d(n) = n + 1, \quad 1 \leq n \leq d + 1, \]
\[ N_d(n) = N_d(n - 1) + N_d(n - d - 1), \quad n > d + 1. \]  
(3.2)

Table 3.2 lists the number of distinct \((d)\) sequences as a function of the sequence length \(n\) with the minimum runlength \(d\) as a parameter. The former recursion relation (3.2) has an elementary interpretation. Let \((x_1, x_2, \ldots, x_{n-1})\) be a \((d)\) sequence of length \(n - 1 > d\), then \((x_1, x_2, \ldots, x_{n-1}, 0)\) and \((x_1, x_2, \ldots, x_{n-d-1}, 0^d, 1)\) are also \((d)\) sequences, where \(0^d\) stands for a string of \(d\) consecutive 'zeros'. As an immediate consequence of this observation, we conclude that the number of distinct \((d)\) sequences of length \(n\) is \(N_d(n - 1) + N_d(n - d - 1)\).

**TABLE 3.2 - Number of distinct \((d)\) sequences as a function of the sequence length \(n\) and the minimum runlength \(d\) as a parameter.**

<table>
<thead>
<tr>
<th>(n = )</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
<th>9</th>
<th>10</th>
<th>11</th>
<th>12</th>
<th>13</th>
<th>14</th>
</tr>
</thead>
<tbody>
<tr>
<td>(d = 1 )</td>
<td>8</td>
<td>13</td>
<td>21</td>
<td>34</td>
<td>55</td>
<td>89</td>
<td>144</td>
<td>233</td>
<td>377</td>
<td>610</td>
<td>987</td>
</tr>
<tr>
<td>2</td>
<td>6</td>
<td>9</td>
<td>13</td>
<td>19</td>
<td>28</td>
<td>41</td>
<td>60</td>
<td>88</td>
<td>129</td>
<td>189</td>
<td>277</td>
</tr>
<tr>
<td>3</td>
<td>5</td>
<td>7</td>
<td>10</td>
<td>14</td>
<td>19</td>
<td>26</td>
<td>36</td>
<td>50</td>
<td>69</td>
<td>95</td>
<td>131</td>
</tr>
<tr>
<td>4</td>
<td>5</td>
<td>6</td>
<td>8</td>
<td>11</td>
<td>15</td>
<td>20</td>
<td>26</td>
<td>34</td>
<td>45</td>
<td>60</td>
<td>80</td>
</tr>
<tr>
<td>5</td>
<td>5</td>
<td>6</td>
<td>7</td>
<td>9</td>
<td>12</td>
<td>16</td>
<td>21</td>
<td>27</td>
<td>34</td>
<td>43</td>
<td>55</td>
</tr>
</tbody>
</table>

When \(d = 0\), we simply find that \(N_d(n) = 2N_d(n - 1)\), or in other words, when there is no restriction at all, the number of combinations doubles when a bit is added, which is, of course, a well-known result.

The numbers \(N_d(n)\) are 1, 2, 3, 5, 8, 13, ..., where each number is the sum of its two predecessors. These numbers are called *Fibonacci numbers*, after the Italian mathematician who discovered that the number of rabbits multiplies in Fibonacci rhythm. The ratio of two successive Fibonacci numbers \(N(n + 1)/N(n)\) approaches for large \(n\) the golden ratio \(g = (1 + \sqrt{5})/2\), which is easily verified. Similarly, the numbers \(N_d(n), \ d > 1\), are called **generalized Fibonacci numbers**.

The number of \((dk)\) sequences of length \(n\) can be found in a similar fashion. Let \(N(n)\) denote the number of \((dk)\) sequences of length \(n\). (For the sake of simplicity in notation no subscript is used in this case). Define

\[
N(n) = 0, \quad n < 0, \\
N(0) = 1.
\]  
(3.3)

The number of \((dk)\) sequences of length \(n\) is given by
\( N(n) = n + 1, \ 1 \leq n \leq d + 1, \)
\( N(n) = N(n - 1) + N(n - d - 1), \ d + 1 \leq n \leq k, \)
\[
N(n) = d + k + 1 - n + \sum_{i=d}^{k} N(n - i - 1), \ k < n \leq d + k,\tag{3.4}
\]
\[
N(n) = \sum_{i=d}^{k} N(n - i - 1), \ n > d + k.
\]

The \( k \)-limited case, \( d = 0 \), can be derived as a special case of the general \( dk \) case. If \( N_k(n) \) denotes the number of \( (k) \) sequences of length \( n \), the following recursion relations can be written down:
\[
N_k(n) = 2^n, \ 0 < n \leq k,\tag{3.5}
\]
\[
N_k(n) = \sum_{i=1}^{k+1} N_k(n - i), \ n > k.
\]

It is easily seen that \( N_{k-1}(n), \ n = 1, 2, \ldots, \) is the sequence of Fibonacci numbers.

### 3.3 Asymptotic Information Rate

An encoder translates arbitrary user (or source) information into, in this particular case, a sequence that satisfies given \( d \) and \( k \) constraints. On the average \( m \) source symbols are translated into \( n \) channel symbols. The quotient \( R = m/n, \ m < n \), is called the code rate. What is the maximum value of \( R \) that can be attained for some specified values of the minimum and maximum runlength \( d \) and \( k \)? The answer, as discussed in Chapter 2, was provided by Shannon [1]; the maximum value of \( R \) that can be achieved is called the capacity of a \( (dk) \) code. The capacity, or asymptotic information rate, of \( (dk) \) sequences, denoted by \( C(d,k) \), defined as the number of information bits per channel bit that can maximally be carried by the \( (dk) \) sequences, on average, is governed by the specified constraints and is given by
\[
C(d,k) = \lim_{n \to \infty} \frac{1}{n} \log_2 N(n).\tag{3.6}
\]

For notational convenience we restrict ourselves for the time being to \( (d) \) sequences. Eq. (3.6) requires an explicit formula of the number of sequences \( N(n) \) as a function of the sequence length \( n \). The desired expression is most easily obtained by solving the homogeneous
difference equation (3.2). According to (3.2) the number of \(d\) sequences is

\[ N_d(n) = N_d(n-1) + N_d(n-d-1), \quad n > d + 1. \]

Writing \(N_d(n) = z^n\), we obtain the characteristic equation

\[ z^{d+1} - z^d - 1 = 0. \]  \hspace{1cm} (3.7)

Any \(z\) that satisfies the characteristic equation will solve the difference equation. The general solution for \(N_d(n)\) is then a linear combination:

\[ N_d(n) = \sum_{i=1}^{d+1} a_i \lambda_i^n, \]

where \(\lambda_i, \quad i = 1, \ldots, d + 1\), are the roots of (3.7), and \(a_i\) are constants to be chosen to meet the first \((d + 1)\) values of \(N_d(n)\). If \(\lambda = \max \{\lambda_i\}\) is the largest real root of (3.7), then for large \(n\),

\[ N_d(n) \propto \lambda^n. \]

Applying definition (3.6), the asymptotic information rate (or capacity) of \(d\)-constrained sequences, denoted by \(C(d, \infty)\), is

\[ C(d, \infty) = \lim_{n \to \infty} \frac{1}{n} \log_2 N_d(n) = \log_2 \lambda. \]  \hspace{1cm} (3.8)

This is the fundamental result that we need: the quantity \(C(d, \infty)\) supplies the maximum rate possible of any implemented code given the channel constraints.

**Example 3.1:** Let \(d = 1\), then we obtain the characteristic equation

\[ z^2 - z - 1 = 0, \]

with solutions

\[ \lambda_1 = \frac{1}{2} \left(1 + \sqrt{5}\right) \quad \text{and} \quad \lambda_2 = \frac{1}{2} \left(1 - \sqrt{5}\right). \]

After some rearrangement we obtain, quite surprisingly, an explicit formula, discovered by A. de Moivre in 1718 and proved some years later by Nicolas Bernoulli:
Runlength-Limited Sequences

\[
N_1(n) = \frac{1}{\sqrt{5}} \left[ \frac{1}{2} (1 + \sqrt{5}) \right]^n - \frac{1}{\sqrt{5}} \left[ \frac{1}{2} (1 - \sqrt{5}) \right]^n
= \frac{1}{\sqrt{5}} \{ g^n - (-g)^{-n} \}. \tag{3.9}
\]

So that \( \lambda = g = (1 + \sqrt{5})/2 \), and

\[
C(1, \infty) = \log_2 \frac{1 + \sqrt{5}}{2} \approx 0.694.
\]

Some further results of computations, which are obtained by numerical methods, are collected in Table 3.3.

\begin{table}[h]
\centering
\begin{tabular}{|c|c|c|}
\hline
\textbf{d} & \textbf{C(d,\infty)} & \textbf{(d + 1)C(d,\infty)} \\
\hline
1 & 0.694 & 1.388 \\
2 & 0.551 & 1.654 \\
3 & 0.465 & 1.860 \\
4 & 0.406 & 2.028 \\
\hline
\end{tabular}
\caption{Capacity and density ratio \( DR \) versus minimum runlength \( d \).}
\end{table}

The quantity \( DR \), called \textit{density ratio}, sometimes called \textit{packing density}, which expresses the minimum physical distance between consecutive transitions of an RLL sequence, is defined as

\[
DR = (1 + d)R, \tag{3.10}
\]

where \( R \) is the rate of the RLL code. Obviously, \( R = C(d,k) \) when the RLL sequence is maxentropic. It can be seen in the table that an increase of the density ratio can be obtained at the expense of decreased code rate. It can even be shown that the density ratio \( DR \) can be made arbitrarily large by choosing the minimum runlength \( d \) sufficiently large. This follows from

\[
DR = (d + 1) \log_2 \lambda.
\]

The root \( \lambda \) of (3.7) satisfies

\[
\left( \frac{(1 - \varepsilon)d}{\log_2 d} \right)^{1/d} \leq \lambda \leq \left( \frac{(1 + \varepsilon)d}{\log_2 d} \right)^{1/d}.
\]
for large \( d \). Thus \( DR \) grows like a constant times \( \log d \), see also Table 3.3. It should be appreciated that codes with a larger value of \( d \), and thus a lower rate, provide an increasingly difficult trade-off between the detection window and the density ratio in applications with very high information density and data rates. Some of the parameters involved in the system trade-off are discussed in Chapter 1.

In similar vein to the case of \( d \)-constrained sequences, it is possible to derive the capacity \( C(d,k) \) of \((dk)\) sequences. Sequences that meet prescribed \( dk \) constraints may be thought to be composed of phrases of length (duration) \( j + 1, d \leq j \leq k \), denoted by \( T_{j+1} \), of the form \( \{10^4, 10^{k-1}, \ldots, 10^j, \ldots, 10^4\} \), where \( 0 \) stands for a sequence of \( j \) consecutive 'zeros'. As an immediate consequence of (2.24), page 35, the characteristic equation of \((dk)\) sequences is (for finite \( k \))

\[
\sum_{i=0}^{(d+1)} z^{-i} = 0,
\]

or

\[
-z^{k+2} - z^{k+1} - z^{k-d+1} + 1 = 0. \tag{3.11}
\]

Table 3.4 lists the capacity \( C(d,k) \) versus the runlength parameters \( d \) and \( k \).

<table>
<thead>
<tr>
<th>( k )</th>
<th>( d = 0 )</th>
<th>( d = 1 )</th>
<th>( d = 2 )</th>
<th>( d = 3 )</th>
<th>( d = 4 )</th>
<th>( d = 5 )</th>
<th>( d = 6 )</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>.8791</td>
<td>.4057</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>.9468</td>
<td>.5515</td>
<td>.2878</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>.9752</td>
<td>.6174</td>
<td>.4057</td>
<td>.2232</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>.9881</td>
<td>.6509</td>
<td>.4650</td>
<td>.3218</td>
<td>.1823</td>
<td></td>
<td></td>
</tr>
<tr>
<td>6</td>
<td>.9942</td>
<td>.6690</td>
<td>.4979</td>
<td>.3746</td>
<td>.2669</td>
<td>.1542</td>
<td></td>
</tr>
<tr>
<td>7</td>
<td>.9971</td>
<td>.6793</td>
<td>.5174</td>
<td>.4057</td>
<td>.3142</td>
<td>.2281</td>
<td>.1335</td>
</tr>
<tr>
<td>8</td>
<td>.9986</td>
<td>.6853</td>
<td>.5293</td>
<td>.4251</td>
<td>.3432</td>
<td>.2709</td>
<td>.1993</td>
</tr>
<tr>
<td>9</td>
<td>.9993</td>
<td>.6888</td>
<td>.5369</td>
<td>.4376</td>
<td>.3620</td>
<td>.2979</td>
<td>.2382</td>
</tr>
<tr>
<td>10</td>
<td>.9996</td>
<td>.6909</td>
<td>.5418</td>
<td>.4460</td>
<td>.3746</td>
<td>.3158</td>
<td>.2633</td>
</tr>
<tr>
<td>11</td>
<td>.9998</td>
<td>.6922</td>
<td>.5450</td>
<td>.4516</td>
<td>.3833</td>
<td>.3282</td>
<td>.2804</td>
</tr>
<tr>
<td>12</td>
<td>.9999</td>
<td>.6930</td>
<td>.5471</td>
<td>.4555</td>
<td>.3894</td>
<td>.3369</td>
<td>.2924</td>
</tr>
<tr>
<td>13</td>
<td>.9999</td>
<td>.6935</td>
<td>.5485</td>
<td>.4583</td>
<td>.3937</td>
<td>.3432</td>
<td>.3011</td>
</tr>
<tr>
<td>14</td>
<td>.9999</td>
<td>.6938</td>
<td>.5495</td>
<td>.4602</td>
<td>.3968</td>
<td>.3478</td>
<td>.3074</td>
</tr>
<tr>
<td>15</td>
<td>.9999</td>
<td>.6939</td>
<td>.5501</td>
<td>.4615</td>
<td>.3991</td>
<td>.3513</td>
<td>.3122</td>
</tr>
<tr>
<td>( \infty )</td>
<td>1.000</td>
<td>.6942</td>
<td>.5515</td>
<td>.4650</td>
<td>.4057</td>
<td>.3620</td>
<td>.3282</td>
</tr>
</tbody>
</table>
3.3.1 State-Transition Matrix Description

![State-transition diagram for a \((d,k)\) sequence. Transmission of a 'zero' takes the sequence from state \(\sigma_i\) to state \(\sigma_{i+1}\), \(i \leq k\). A 'one' may be transmitted only when the machine occupies states \(\sigma_{d+1}, ..., \sigma_{k+1}\). The machine returns to state \(\sigma_1\) after transition of a 'one'.]

There is an alternative useful technique to derive the channel capacity, which is based on the representation of the \(d\) and \(k\) constraints by a finite-state sequential machine. Figure 3.1 illustrates a possible state-transition diagram. There are \((k + 1)\) states which are denoted by \(\{\sigma_1, ..., \sigma_{k+1}\}\). Transmission of a 'zero' takes the sequence from state \(\sigma_i\) to state \(\sigma_{i+1}\). A 'one' may be transmitted only when the machine occupies states \(\sigma_{d+1}, ..., \sigma_{k+1}\). The adjacency or connection matrix, which gives the number of ways (paths) of going (in one step) from state \(\sigma_i\) to state \(\sigma_j\), is given by the \((k + 1) \times (k + 1)\) array \(D\) with entries \(d_{ij}\), where

\[
\begin{align*}
d_{ij} &= 1, \quad i \geq d + 1, \\
d_{ij} &= 1, \quad j = i + 1, \\
d_{ij} &= 0, \quad \text{otherwise.}
\end{align*}
\]

For example, the connection matrix for \((d,k) = (1,3)\) is

\[
D = \begin{bmatrix}
0 & 1 & 0 & 0 \\
1 & 0 & 1 & 0 \\
0 & 0 & 1 & 0 \\
0 & 0 & 0 & 0
\end{bmatrix}.
\]

The above representation is related to the input-restricted noiseless channel studied by Shannon (see Chapter 2). The finite-state machine model allows us to compute the capacity, and it is also very helpful to compute the number of sequences that start and end in certain states.
Applying (2.18), page 31, we find that the capacity of a Markov source that emits \(dk\)-constrained sequences is
\[
C(d,k) = \log_2 \lambda,
\]
where \(\lambda\) is the largest real root of the characteristic equation
\[
\det[D - zI] = 0,
\]
and \(I\) is the identity matrix. It is left for the reader as an exercise to demonstrate that the former equation coincides with (3.11), page 44.

### 3.4 Statistical Properties of Maxentropic Runlength-Limited Sequences

In a maxentropic RLL sequence, the runlength of length \(T_i\) has probability of occurrence
\[
Pr(T_i) = \lambda^{-i}, \quad i = d + 1, \ldots, k + 1,
\]
where \(\lambda\) is the largest real root of (3.11).

To deduce the transition probabilities \(q_{ij}\) that maximize the entropy of the \((k + 1)\)-state runlength-limited source when the connection matrix \(D\) is given, one must determine the right eigenvector \(\hat{\mathbf{v}} = (\hat{v}_1, \hat{v}_2, \ldots, \hat{v}_N)^T\) that satisfies (see Chapter 2):
\[
D\hat{\mathbf{v}} = \lambda\hat{\mathbf{v}}.
\]

The right eigenvector is
\[
\hat{\mathbf{v}} = (1, \lambda, \lambda^2, \ldots, \lambda^d, (\lambda^{d+1} - 1), (\lambda^{d+2} - \lambda - 1), \ldots, (\lambda^k - \lambda^{k-d} - \ldots - \lambda - 1))^T.
\]

The joint probability \(q_{ij}\) of a transition from state \(i\) to \(j\) of the maxentropic Markov source is
\[
q_{ij} = \frac{1}{\lambda} d_{ij} \frac{\hat{v}_j}{\hat{v}_i}, \quad i,j = 1,2,\ldots,k + 1.
\]

The matrix \(Q\) is stochastic, that is \(\sum q_{ij} = 1\).

The probability \(\pi_i\) associated with state \(\sigma_i\) is given by \(\hat{\mathbf{v}}_i\hat{\mathbf{u}}_i\), where \(\hat{\mathbf{v}}\) and \(\hat{\mathbf{u}}\) are the right and left eigenvectors for \(D\) with eigenvalue \(\lambda\), respectively, and \(\sum \pi_i = 1\). The left eigenvector \(\hat{\mathbf{u}}\), is given by
\[
\hat{\mathbf{u}} = (\lambda^k, \lambda^{k-1}, \ldots, 1),
\]
and the stationary probability $\pi_i$ is

$$\rho \pi_i = \begin{cases} 
\lambda^k, & 1 \leq i \leq d + 1 \\
\lambda^k - \sum_{j=1}^{i-1-d} \lambda^{k-d-j}, & d + 2 \leq i \leq k + 1, 
\end{cases} \quad (3.19)$$

where the normalization constant $\rho$ is chosen to satisfy

$$\sum \pi_i = 1.$$ 

Maxentropic RLL codes are symmetrical with respect to reversal. Reversing the directions of all edges leads to a state-transition diagram whose adjacency matrix is the transpose of $D$.

**FIGURE 3.2** - Reversed state-transition diagram. The figure is the same as Figure 3.1, but now the arrows are pointing in the opposite direction.

The reversed state-transition diagram describes the same set of runlength parameters as the original, the reverse transition diagram is shown in Figure 3.2. The reversed diagram provides a simpler way to compute the runlength distribution of a maxentropic RLL sequence than the original diagram. The probability of runlength $g$, $d \leq g \leq k$, 'zeros' is just the probability $\tilde{q}_{1g}$ from state $\sigma_1$ to state $\sigma_g$. Using the formulas $\tilde{q}_g = \frac{1}{\lambda} \hat{u}_g \hat{u}_1$ and $\hat{u}_1 = \lambda^{k-d-1}$ from above, we obtain $\tilde{q}_{1g} = \lambda^{-g}$. The probability of a phrase of length $g$ consisting of a 'one' followed by $g-1$ 'zeros' starting and ending at the state labelled 1 is just $\lambda^{-g}$. 
3.4.1 Spectrum of Maxentropic RLL Sequences

If it is assumed that a transmitter emits the phrases $T_j$ independently with probability $Pr(T_j)$, then the power spectral density function of the corresponding RLL sequence is given by [2]

$$H(\omega) = \frac{1}{\bar{T} \sin^2 \omega/2} \frac{1 - |G(\omega)|^2}{|1 + G(\omega)|^2},$$

(3.20)

where

$$G(\omega) = \sum_{l=d+1}^{k+1} Pr(T_l) e^{i\omega l}$$

(3.21)

and

$$\bar{T} = \sum_{l=d+1}^{k+1} lPr(T_l).$$

(3.22)

The runlengths of a maxentropic sequence follow, as earlier argued, a truncated geometric distribution with parameter $\lambda$:

$$Pr(T_l) = \lambda^{-l}, \quad l = d + 1, d + 2, \ldots, k + 1,$$

(3.23)

whence

$$\bar{T} = \sum_{l=d+1}^{k+1} lPr(T_l) = \sum_{l=d+1}^{k+1} l\lambda^{-l}.$$  

(3.24)

Substitution of the distribution provides a straightforward method of determining the spectrum of maxentropic RLL sequences. Figure 3.3 depicts the spectra $H(\omega)$ of some maxentropic $(d)$ sequences for various values of the minimum runlength $d$.

We may observe the following characteristics: maxima occur at non-zero frequency, and the spectra exhibit a more pronounced peak with increasing $d$. The energy in the low-frequency range diminishes with decreasing minimum runlength $d$. The effects on the spectra of a reduction of the maximum runlength can be seen in Figure 3.4. The figure depicts the spectrum of $(d = 2)$ sequences with the maximum runlength $k$ as a parameter.
FIGURE 3.3 - Power density function versus frequency of maxentropic runlength-limited sequences. Both the frequency scale and the pulse lengths of the RLL sequences are normalized in such a way that the information rate of all sequences is fixed at 1 bit/s. The vertical axis is scaled for unity total power.

FIGURE 3.4 - Power density function versus frequency of maxentropic $d$-constrained runlength-limited sequences with $d = 2$ and some selected $k$ values.
3.5 Practical Coding Schemes

In the previous sections of this chapter, properties, such as capacity and spectra, of maxentropic RLL sequences are treated. Thus we have now acquired an information-theoretical knowledge on the key aspects of RLL sequences. In the present section we take a closer look at the techniques that are available to produce runlength-limited sequences in a practical manner. It is most important that this be done as efficiently as possible within some practical considerations. A good algorithm realizes a code rate that is close to the capacity of the constrained sequences, uses a simple implementation, and avoids the propagation of errors in the process of decoding. The previous sections on maxentropic RLL sequences provide the cardinal limits. Codes that have found application outside the laboratory apply to runlength-limited codes with parameters \( d = 0, 1, \) and 2, which is, of course, reflected in the survey.

### 3.5.1 Fixed-Length Binary \((d,k)\) Codes

One approach that has proved very successful for the conversion of arbitrary source information into constrained sequences is the one constituted by block codes. The source sequence is partitioned into blocks of length \( m \), and under the code rules such blocks are mapped onto words of \( n \) channel symbols. A code may be state-dependent, in which case the choice of the codeword used to represent a given binary source block is a function of the channel or encoder state, or the code may be state independent. State independence implies that codewords can be freely concatenated without violating the sequence constraints. This additional restriction leads, in general, to codes that are longer than state-dependent codes for given bit-per-symbol value. In some instances, state independence may yield advantages in error propagation limitation. The EFM code (see Chapter 1) is a proper representative of a code with limited error propagation. State-independent decoding may be achieved for any fixed-length \((d,k)\) code, as will be indicated later. To clarify this concept we have written down a simple illustrative case of a \((1, \infty)\) code.

The codeword assignment of Table 3.5 provides a simple block code that converts blocks of length \( m = 3 \) onto codewords of length \( n = 5 \). In order to merge the codewords without violating the \( d = 1 \) constraint, we have selected the first symbol of the codewords to be a 'zero'. There are exactly eight codewords of length \( n - 1 = 4 \) that meet the specified
Runlength-Limited Sequences 51

d = 1 constraint (see also Table 3.2), so that a one-to-one mapping between source words and codewords is possible.

<table>
<thead>
<tr>
<th>source</th>
<th>output</th>
</tr>
</thead>
<tbody>
<tr>
<td>0 000</td>
<td>00000</td>
</tr>
<tr>
<td>1 001</td>
<td>00001</td>
</tr>
<tr>
<td>2 010</td>
<td>00010</td>
</tr>
<tr>
<td>3 011</td>
<td>00100</td>
</tr>
<tr>
<td>4 100</td>
<td>00101</td>
</tr>
<tr>
<td>5 101</td>
<td>01000</td>
</tr>
<tr>
<td>6 110</td>
<td>01001</td>
</tr>
<tr>
<td>7 111</td>
<td>01010</td>
</tr>
</tbody>
</table>

The code rate is \( m/n = 3/5 < C(1, \infty) = 0.69 \). The code efficiency, designated by \( \eta \), expressed as the quotient of code rate and capacity of the maxentropic sequence with the same runlength constraints is

\[
\eta = \frac{R}{C(d,k)} \approx 0.6 \approx 0.86. \quad (3.25)
\]

This, actually, demonstrates that good efficiencies are feasible with practicable embodiments. That this example is not untypical will be demonstrated in the remainder of this chapter. The decoding of the received codewords can in fact be achieved in a very simple fashion: the decoder skips the first symbol of each received codeword, and, using a look-up table, it maps the four remaining codeword symbols onto the retrieved source word. The codewords have been allotted to the source words in an arbitrary fashion, and, evidently, other assignments might be chosen instead. A different map may aim to simplify the implementation of the look-up tables for encoding and decoding. The case under study is so simple that implementation considerations are not worth the effort, but when the codebook is larger, a detailed study might save many logic gates.

It is quite straightforward to generalize the preceding implementation example to encoders that generate sequences with an arbitrary value of the minimum runlength. To that end, choose some appropriate codeword length \( n \). Set the first \( d \) symbols of each codeword to 'zero'. The number of codewords that meet the given runlength constraints is...
conditions is \( N_d(n - d) \), which can be computed with (3.2) or by using Table 3.2.

A maximum runlength constraint can be incorporated in the code rules in a straightforward manner. For instance, in the \( d = 1 \) code previously described, the first codeword symbol is preset to 'zero'. If, however, the last symbol of the preceding codeword and the second symbol of the actual codeword to be conveyed are both 'zero', then the first codeword symbol can be set to 'one' without violating the \( d = 1 \) channel constraint. This extra rule, which governs the selection of the two merging bits, the *merging rule*, can quite smoothly be implemented with some extra hardware. It is readily conceded that with this additional 'merging' rule the \((d,k) = (1, \infty)\) code, presented in Table 3.5, turns into a \((d,k) = (1,6)\) code. The code efficiency is now, as can be verified with Table 3.4: \( \eta = 0.6/C(1,6) = 0.6/0.669 \approx 0.897 \). The process of decoding is exactly the same as for the simple \((d,k) = (1, \infty)\) code, since the first bit, the 'merging' bit, was skipped anyway. The \((d,k) = (1,6)\) code is a good illustration of the state-dependent encoding principle (the actual codeword transmitted depends on the previous codeword) and state-independent decoding (the retrieved source word does not depend on previous codewords or the channel state).

**Modified Frequency Modulation**

Modified Frequency Modulation (MFM), a rate \( \frac{1}{2}, d = 1, k = 3 \) code has proved very popular from the viewpoint of simplicity and ease of implementation, and has become a *de facto* industry standard in flexible and 'Winchester'-technology disk drives. MFM is essentially a block code of length \( n = 2 \) with a simple merging rule when the NRZI notation is employed. The MFM encoding table is shown in Table 3.6. The symbol indicated with 'x' is set to 'zero' if the preceding symbol is 'one' else it is set to 'one'. It can be verified that this construction yields a maximum runlength \( k = 3 \).

<table>
<thead>
<tr>
<th>Source</th>
<th>Output</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>x0</td>
</tr>
<tr>
<td>1</td>
<td>01</td>
</tr>
</tbody>
</table>

MFM has high efficiency, \( \eta \approx 0.5/0.5515 \), or approximately 91 percent. A graphical representation of the finite-state machine underlying
the MFM code, using the NRZI notation rules, is pictured in Figure 3.5. The labelled edges emanating from a state define the encoding rule, and the state in which an edge terminates indicates the state, or coding rule to use next. The state A represents the condition that the previous channel bit was a ‘zero’, while the state B indicates that the previous channel bit was a ‘one’. Decoding of the MFM code is simply accomplished by discarding the redundant first bit in each received 2-bit block.

3.5.2 Fixed-Length Codes of Minimum Length

As was already pointed out, the $d$ and $k$ constraints define a number of channel states. The crucial problem for the creation of fixed-length codes of minimum length is to find a subset of the channel states, referred to as principal states, of which there exist a sufficient number of sequences of length $n$ terminating at other principal states. The existence of a set of principal states is a necessary and sufficient condition for the existence of a code with the specified rate and codeword length. Franaszek [3] developed a recursive search technique for determining the existence of a set of principal states through operations on the connection matrix. The subsequent procedure decides whether there exists a set of principal states for the specified parameters.

Let the codeword length $n$ and the source word length $m$ be given. The specified channel constraints $d$ and $k$ define a set of channel states denoted by $\Sigma = \{\sigma\}$. Let further $\Sigma'$ be the set of states that have not been eliminated and $\sigma_i \in \Sigma'$ a state to be tested. The number $\psi(\sigma_i, \Sigma')$ of $(dk)$ sequences of length $n$ permitted from $\sigma_i$ and terminating in a state $\sigma_j \in \Sigma'$ is given by
\[ \psi(\sigma_i, \Sigma^*) = \sum_{j \in V} [D]^n_{ij}, \quad V = \{j : \sigma_j \in \Sigma^*\}, \quad (3.26) \]

where \([D]^n_{ij}\) denotes the entries of the \(n\)th power of the connection matrix \(D\). If \(\psi(\sigma_i, \Sigma^*) < 2^n\), \(\sigma_i\) is eliminated from \(\Sigma^*\). Starting with \(\Sigma^* = \Sigma\), the routine is continued until either all states have been eliminated, or the routine goes through a complete cycle of remaining states without further elimination. In the latter case, we know that for any \(\sigma_i \in \Sigma^*\),

\[ \psi(\sigma_i, \Sigma^*) \geq 2^n, \]

thus \(\Sigma^*\) is the set of principal states.

The following illustrations have been chosen to clarify some of the points dealt with in the preceding sections.

**Example 3.2:** We examine here the implementation of a \((d = 1, k = 3)\) code. Table 3.4 indicates that a code rate 1/2 represents 90 percent of the channel capacity. Therefore, let \(m = 1\) and \(n = 2\). The matrix \(D^2\) is

\[
D^2 = \begin{bmatrix}
1 & 0 & 1 & 0 \\
1 & 1 & 0 & 1 \\
1 & 1 & 0 & 0 \\
0 & 1 & 0 & 0
\end{bmatrix}.
\quad (3.27)
\]

Use of the recursive search algorithm indicates that state \(\sigma_4\) has to be deleted (the row sum for row four is only one). We eliminate row four and column four. The \(3 \times 3\) submatrix so obtained has row sums exactly two. Thus, the principal states are \(\sigma_1\), \(\sigma_2\), and \(\sigma_3\). The codewords available for encoding associated with the principal states are

\[
W(\sigma_1) = \begin{cases}
01 \\
00
\end{cases}
\]

\[
W(\sigma_2) = \begin{cases}
01 \\
10
\end{cases}
\]

\[
W(\sigma_3) = \begin{cases}
01 \\
10
\end{cases}.
\quad (3.28)
\]

A state-independently decodable code may be constructed with the assignments
After some rearrangement we obtain the following simplified codebook:

<table>
<thead>
<tr>
<th>source</th>
<th>output</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>x0</td>
</tr>
<tr>
<td>1</td>
<td>01</td>
</tr>
</tbody>
</table>

A comparison with Table 3.6 reveals that this is the MFM code.

**Example 3.3:** Let $d = 2$ and $k = \infty$. The connection matrix $D$ is given by

$$D = \begin{bmatrix} 0 & 1 & 0 \\ 0 & 0 & 1 \\ 1 & 0 & 1 \end{bmatrix}.$$ 

The capacity of the channel is (see Table 3.3, page 43)

$$C(d = 2, \infty) \approx 0.551.$$ 

It is quite plausible to suppose that fairly short codes exist with a rate $1/2$, and we proceed to show that such a fixed-length code indeed exists. Using the recursive search procedure previously described, one can prove that the shortest fixed-length code with rate $1/2$ has a codeword length of 14. Note that

$$D^{14} = \begin{bmatrix} 41 & 28 & 60 \\ 60 & 41 & 88 \\ 88 & 60 & 129 \end{bmatrix}.$$ 

All three states are principal states. We may easily verify that
\[
\sum_{j=1}^{3} [D]_{ij}^{14} = 129 \\
\sum_{j=1}^{3} [D]_{2j}^{14} = 189 \\
\sum_{j=1}^{3} [D]_{3j}^{14} = 277.
\]

So that indeed in any state more than \(2^n = 2^7 = 128\) sequences which comply with the channel restrictions start and end. To form a rate 7/14 code, we may choose any 128 of the possible 129 sequences and allocate the source words to the codewords.

Perusal of the \(D^{14}\) matrix reveals that the lower-right element is greater than 128. So that, in fact, only one terminal state, namely \(\sigma_3\), suffices for encoding. Apparently, this particular code can be state-independently encoded and decoded (all sequences start and end in state \(\sigma_3\)). Actually, this outcome is not surprising, for, when we take a look at Table 3.2, we notice there are exactly 129 \((d = 2)\) sequences of length 12. The addition of two merging bits, which are preset to 'zero', completes the design of a rate 7/(12 + 2), \((d = 2, k = \infty)\) block code.

It is clear that the 14-bit codewords can be cascaded in any order without violating the prescribed runlength restrictions.

The shortest fixed-length codes of the specified rate for a selection of \((d,k)\) combinations were computed with the recursive search procedure; the results are collected in Table 3.7. The parameter \(L = |\Sigma_p|\) denotes the minimum number of principal states.

Tables 3.8a and 3.8b list the smallest codeword length that is possible for the specified rate and \(dk\) constraints with fixed-length codewords. As can be seen from the tables, the codeword length required increases when the maximum runlength \(k\) is reduced, or in other words, when the actual rate of the code approaches the channel capacity for the specified conditions. We draw attention to the fact that the minimum codeword lengths of the fixed-length \((d = 1, k = 7)\), \(R = 2/3\) code and the \((d = 2, k = 7)\), \(R = 1/2\) code are 33 and 34, respectively. For these specific
TABLE 3.7 - Shortest fixed-length block codes of given bit-per-symbol values for a selection of $(d,k)$ constraints.

| $d$ | $k$ | $m$ | $n$ | $|\Sigma_p|$ | $\eta = R/C(d,k)$ |
|-----|-----|-----|-----|-------------|------------------|
| 0   | 1   | 3   | 5   | 1           | 0.864            |
| 0   | 2   | 4   | 5   | 2           | 0.910            |
| 0   | 3   | 9   | 10  | 2           | 0.951            |
| 1   | 3   | 1   | 2   | 3           | 0.907            |
| 1   | 5   | 6   | 10  | 4           | 0.922            |
| 2   | 5   | 4   | 10  | 4           | 0.860            |
| 2   | 8   | 11  | 22  | 7           | 0.945            |
| 2   | 11  | 8   | 16  | 8           | 0.917            |
| 3   | 7   | 46  | 115 | 7           | 0.986            |
| 3   | 11  | 8   | 20  | 6           | 0.886            |
| 4   | 9   | 9   | 27  | 9           | 0.921            |
| 4   | 14  | 12  | 33  | 8           | 0.916            |
| 5   | 12  | 9   | 30  | 8           | 0.890            |
| 5   | 17  | 15  | 45  | 15          | 0.937            |

TABLE 3.8a - Shortest fixed-length rate $= 2/3$, $d = 1$ code for a selection of $k$ constraints.

<table>
<thead>
<tr>
<th>$k$</th>
<th>$n$</th>
<th>$\eta = R/C(d,k)$</th>
</tr>
</thead>
<tbody>
<tr>
<td>11</td>
<td>18</td>
<td>0.963</td>
</tr>
<tr>
<td>10</td>
<td>21</td>
<td>0.965</td>
</tr>
<tr>
<td>9</td>
<td>21</td>
<td>0.968</td>
</tr>
<tr>
<td>8</td>
<td>24</td>
<td>0.973</td>
</tr>
<tr>
<td>7</td>
<td>33</td>
<td>0.981</td>
</tr>
<tr>
<td>6</td>
<td>165</td>
<td>0.996</td>
</tr>
</tbody>
</table>

TABLE 3.8b - Shortest fixed-length rate $= 1/2$, $d = 2$ code for a selection of $k$ constraints.

<table>
<thead>
<tr>
<th>$k$</th>
<th>$n$</th>
<th>$\eta = R/C(d,k)$</th>
</tr>
</thead>
<tbody>
<tr>
<td>13</td>
<td>14</td>
<td>0.912</td>
</tr>
<tr>
<td>10</td>
<td>16</td>
<td>0.923</td>
</tr>
<tr>
<td>9</td>
<td>18</td>
<td>0.931</td>
</tr>
<tr>
<td>8</td>
<td>22</td>
<td>0.945</td>
</tr>
<tr>
<td>7</td>
<td>34</td>
<td>0.966</td>
</tr>
</tbody>
</table>
cases, a design with variable-length codewords (see later) provides shorter word lengths with reduced error propagation.

### 3.5.3 Fixed-Length Codes Based on (dklr) Sequences

The previous construction technique has the obvious drawback that it requires a large amount of look-up tables for encoding. As a matter of fact, the number of look-up tables equals the number of principal states, which, if the maximum runlength $k$ is large, can be quite prohibitive. The following systematic construction technique constituted by (dklr) sequences has the advantage that only one look-up table is needed for the generation of the (dklr) sequences, plus some logic to determine the merging bits used to cascade the (dklr) sequences. Essentially, the technique is a generalization of the EFM code which is discussed in Chapter 1.

A (dklr) sequence is a (dk) sequence with two additional constraints:

- **l** constraint: the number of consecutive leading 'zeros' of the sequence, that is, the number of 'zeros' preceding the first 'one', is at most $l$,
- **r** constraint: the number of consecutive trailing 'zeros' of the sequence, that is, the number of 'zeros' succeeding the last 'one', is at most $r$.

The additional constraints on the number of leading and trailing 'zeros' allow a more efficient merging of the sequences than provided by the technique of Tang and Bahl.

As already explained, the (dklr) sequences of length $n$ cannot in general be cascaded without violating the $dk$ constraint at the codeword boundaries. Inserting a number $\beta$ of merging bits between adjacent $n$-sequences makes it possible to preserve the $d$ and $k$ constraints for the cascaded output sequence. A little thought will make it clear that the (dk) sequences require $\beta = d + 2$, $d > 0$, merging bits, whereas only $\beta = d$ merging bits are required for (dklr) sequences, provided that $l$ and $r$ are suitably chosen. We shall now provide two constructions of fixed-length codes with merging rules of increasing complexity and efficiency.

**Construction 1:** Choose $d$, $k$, $r$, $l$, and $n$ such that $r + d + l \leq k$ and let $\beta = d$. Then the (dklr) sequences of length $n$ can be freely cascaded without violating the specified $d$ and $k$ constraints if the $\beta$ merging bits are all set to 'zero'. In other words, the code is self-concatenable.
The rate-ineffectiveness of the $d$ merging bits which are all set to 'zero' can be improved by the following construction.

**Construction 2:** Choose $d$, $k$, and $n$ such that $k \geq 2d$. Let $r = l = k - d$ and $\beta = d$. Then the $(dklr)$ sequences of length $n$ can be cascaded without violating the specified $d$ and $k$ constraints if the $\beta$ merging bits are governed by the following rules, which can easily be implemented. Let an $n$-sequence end with a run of $s$ 'zeros' ($s \leq r$) while the next sequence starts with $t$ ($t \leq l$) leading 'zeros'. Table 3.9 shows the merging rule for the $\beta = d$ merging bits.

In order to demonstrate the efficiency of the codes based on Constructions 1 and 2 we will consider some specific cases. For $m = 8$ and for $d = 1, \ldots, 4$ and $k = 2d, \ldots, 20$ we have selected $n$ in such a way that the information rate $R$ was maximized.

### TABLE 3.9 - Merging rules of $(dklr)$ sequences.

<table>
<thead>
<tr>
<th>$s, t$</th>
<th>Merging bits</th>
</tr>
</thead>
<tbody>
<tr>
<td>$s + t + d \leq k$</td>
<td>$0^d$</td>
</tr>
<tr>
<td>$s + t + d \geq k + 1$</td>
<td></td>
</tr>
<tr>
<td>if $s \leq d$</td>
<td>$0^{d-s}1^{s-1}$</td>
</tr>
<tr>
<td>if $s &gt; d$</td>
<td>$1^{d-1}$</td>
</tr>
</tbody>
</table>

Tables 3.10 and 3.11 give the results for $m = 8$ and $d = 1, 2, 3,$ and 4. In order to limit the length of the tables, we have restricted $k$ and $n$ to those values which maximize the code rate $R$. We note that rates up to 95 percent of the channel capacity $C(d,k)$ can be attained. It will be noticed from the tables that on average there is a slight difference in the code efficiencies obtained by Constructions 1 and 2, approximately three percent in favour of Construction 2. With the recursive search technique it can be verified that the codes presented in Table 3.11 are of minimum length.

**TABLE 3.10 - Fixed-length block codes based on Construction 1.**

<table>
<thead>
<tr>
<th>$d$</th>
<th>$k$</th>
<th>$n$</th>
<th>$R$</th>
<th>$\eta = R/C(d,k)$</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>7</td>
<td>12</td>
<td>8/13</td>
<td>0.91</td>
</tr>
<tr>
<td>2</td>
<td>17</td>
<td>14</td>
<td>8/16</td>
<td>0.91</td>
</tr>
<tr>
<td>3</td>
<td>14</td>
<td>17</td>
<td>8/20</td>
<td>0.87</td>
</tr>
<tr>
<td>4</td>
<td>18</td>
<td>19</td>
<td>8/23</td>
<td>0.87</td>
</tr>
</tbody>
</table>
TABLE 3.11 - Fixed-length block codes based on Construction 2.

<table>
<thead>
<tr>
<th>d</th>
<th>k</th>
<th>n</th>
<th>R</th>
<th>\eta = R/C(d,k)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>5</td>
<td>12</td>
<td>8/13</td>
<td>0.95</td>
</tr>
<tr>
<td>2</td>
<td>10</td>
<td>14</td>
<td>8/16</td>
<td>0.92</td>
</tr>
<tr>
<td>3</td>
<td>10</td>
<td>17</td>
<td>8/20</td>
<td>0.90</td>
</tr>
<tr>
<td>4</td>
<td>12</td>
<td>19</td>
<td>8/23</td>
<td>0.90</td>
</tr>
</tbody>
</table>

When the code rate \( R \) approaches the channel capacity \( C(d,k) \), fixed-length code constructions have the drawback, as seen in the former examples, that the implementations can be very complex, involving long codewords and potentially large error propagation. In a later section, it will be demonstrated that with a different technique, which is also due to Franaszek, codes can be built that satisfy the same channel constraints with a considerably simpler code implementation.

3.5.4 Enumerative Coding of (d) Sequences

In the previous examples of codes, we have tacitly assumed that a table is used to hold all the codewords, and that we look up the appropriate codeword for the required source sequence. Specifically, when codewords are comparatively long, the method of direct look-up is not very attractive. We can, however, create codewords by an algebraic procedure, called enumerative encoding, which means that there is no need to hold every codeword in a table. Our objective, in this section, is to develop a general enumerative technique for encoding and decoding (d) sequences. Though the procedure can be generalized to the encoding and decoding of (dk) sequences, we confine ourselves for the moment to the simpler case of (d) sequences. To that end, we establish a 1-1 mapping from a set \( T(d,n) \) of (d) sequences of length \( n \) onto a set of integers \( 0,1,\ldots,|T(d,n)|-1 \), where \( |T(d,n)| = N_d(n) \) is the cardinality of \( T(d,n) \). Dropping the parameters, the set \( T \) can be ordered lexicographically as follows: if \( x = (x_{n-1}, \ldots, x_0) \in T \) and \( y = (y_{n-1}, \ldots, y_0) \in T \), then \( y \) is called less than \( x \), in short, \( y < x \), if there exists an \( i, 0 < i < n \), such that \( y_i < x_i \) and \( x_j = y_j, i < j < n \). For example, '00101' < '01010'. The position of \( x \) in the lexicographical ordering of \( T \) is defined to be the rank of \( x \) denoted by \( r(x) \), i.e., \( r(x) \) is the number of all \( y \) in \( T \) with \( y < x \).

**Theorem 3.2:** The rank \( r(x) \) of the binary (d) sequences \( x \in T \) can be calculated according to
Runlength-Limited Sequences

\[ r(x) = \sum_{j=0}^{n-1} N_d(j)x_j, \]  

(3.29)

**Proof:** According to Cover [4] the lexicographic index \( r(x) \) is given by

\[ r(x) = \sum_{j=0}^{n-1} x_j N_d(x_{n-1}, x_{n-2}, \ldots, x_j+1, 0), \]

where \( N_d(x_{n-1}, \ldots, x_k) \) designates the number of elements in set \( T \) for which the first \( n-k \) coordinates are given by \( (x_{n-1}, \ldots, x_k) \). Words with prefix \( (x_{n-1}, \ldots, x_j+1, 0) \) lexicographically precede words with prefix \( (x_{n-1}, \ldots, x_j+1, 1) \). For each \( j \) such that \( x_j = 1 \) we simply count the number of elements of \( T(d,j) = N_d(j) \), which differ from \( x \) in the \( j \)th term and therefore have lower index. By adding all those numbers for \( j = 0, \ldots, n-1 \), we eventually count all the elements in \( T \) of lower index than \( x \).

**Example 3.4:** Consider the set \( T(d,n) = T(1,4) \) of \( (d = 1) \) sequences of length four. We have \( N_1(0) = 1, N_1(1) = 2, N_1(2) = 3, N_1(3) = 5, \) and \( N_1(4) = 8. \) For instance, \( r(1001) = N_1(3) + N_1(0) = 5 + 1 = 6. \) We can now quite readily verify the following transformations:

<table>
<thead>
<tr>
<th>( r(x) )</th>
<th>x</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0000</td>
</tr>
<tr>
<td>1</td>
<td>0001</td>
</tr>
<tr>
<td>2</td>
<td>0010</td>
</tr>
<tr>
<td>3</td>
<td>0100</td>
</tr>
<tr>
<td>4</td>
<td>0101</td>
</tr>
<tr>
<td>5</td>
<td>1000</td>
</tr>
<tr>
<td>6</td>
<td>1001</td>
</tr>
<tr>
<td>7</td>
<td>1010</td>
</tr>
</tbody>
</table>

The inverse function, conversion from a given integer \( l \) to a \( (dk) \) sequence with rank \( l \) can be carried out as follows.
Inverse Algorithm: Let the set $\mathcal{T}(d,n)$ and an integer $I$, $I = 0, \ldots, |\mathcal{T}(d,n)| - 1$, be given. The following algorithm finds $x$ such that $r(x) = I$.

Let $\hat{I} = I$.

for $j = n - 1$ step $-1$ to 0 do
  if $\hat{I} \geq N_d(j)$ then $x_j = 1, \hat{I} = \hat{I} - N_d(j)$ else $x_j = 0$.

The ranking technique described in Theorem 3.2 allows the implementation of channel encoders of moderate complexity. Encoding and decoding is accomplished by a change of the weighting system of binary numbers, i.e., from the usual powers of two representation used in unconstrained binary sequences to the weights $N_d(n - 1), N_d(n - 2), \ldots$. Storage capacity is required for approximately $n$ non-zero weighting coefficients, a full adder, and an accumulator to store the intermediate and final results. The former hardware requirements have to be compared with a look-up table of $2^n$ entries if a non-algebraic method for coding is used. The use of a look-up sets a limit to the codeword length which can be decoded. The limit depends on technology and required bit rate; figures of $n = 12$ to 14 are commonly quoted as typical present-day maxima. The enumerative decoder will contain some elements, e.g., the weighting coefficients look-up table which are virtually identical with the ones in the encoder. The algorithm lends itself very well for a sequential machine implementation. Buffering of the received message will certainly be required whilst encoding and decoding, respectively, are carried out. The encoding circuitry does not require a multiplier because the codewords $x$ are binary valued, and so the multiplications are simple additions. Unfortunately, the reduction in storage requirements is penalized by an increase in the difficulty of implementing the extra 'random' hardware for adding and comparing, which, of course, makes it less attractive when the codewords are relatively small. The serial implementation is not, of course, the only possible one. It would be practicable to do encoding and decoding by means of buffering and a large number of hard-wired adders.

The algorithm for enumerating codewords can be generalized to the encoding and decoding of $(dk)$ sequences and $(dklr)$ sequences, see Appendix A.
3.5.5 Examples of Code Implementation

In this section we shall take a closer look at the various implementations of fixed-length channel codes.

**Rate 8/9 (0,3) Code**

According to Table 3.4, page 44, the capacity of a sequence with no runs of more than three 'zeros' is $C(0,3) \approx 0.947$. Using the same table, we conclude $C(0,2) \approx 0.879 < 8/9$, so that there is no way to construct a rate 8/9 binary code with no runs of more than two 'zeros'. For the specified (0,3) constraints we find

$$D^9 = \begin{bmatrix}
208 & 108 & 56 & 29 \\
193 & 100 & 52 & 27 \\
164 & 85 & 44 & 23 \\
108 & 56 & 29 & 15 \\
\end{bmatrix}.$$  \hspace{1cm} (3.30)

The existence of a fixed-length code can be ascertained with Franaszek's recursive search technique previously described. It is verified without much difficulty that $\sigma_1$ and $\sigma_2$ are the principal states ($\sigma_3$ and $\sigma_4$ are deleted). From any principal state there are at least $293 > 256$ sequences available. This, in principle, concludes the discussion. We can do slightly better. By judiciously discarding a number of potential codewords we arrive at a code in which the pattern $S_y = 100010001$ is not a codeword and also does not occur anywhere in the coded sequence with original or shifted codeword boundaries; thus $S_y$ can be used as a synchronization pattern at selected positions in the data stream to identify format boundaries. A look-up table, or alternatively the enumerative encoding technique, may be used for encoding and decoding; however, in this case a comprehensive word allocation can be obtained to create simple Boolean equations for encoding and decoding. The codeword assignment, which was given by Patel provides simple and inexpensive encoder and decoder logic. The allocation is based on the 'divide and conquer' principle. Any 9-bit codeword is partitioned into three parts: two 4-bit subcodewords and one merging bit. The 8-bit source block is partitioned into two 4-bit digits. The two 4-bit source words are mapped onto the two 4-bit subcodewords using small look-up tables. Some extra hardware is needed for determining the merging bit.
3.5.6 Variable-Length Synchronous Codes

As we have learnt in the preceding section, attempts to increase fixed-length state-dependent code efficiency result in increased codeword length, and thus in rapidly mounting coder and decoder complexity. Variable-length codes, which may combine the advantages of short and long word lengths, are frequently profitable in terms of hardware complexity. Variable-length codes offer the possibility of using short words more frequently than those of longer lengths. This often permits a marked reduction in coder and decoder complexity relative to a fixed-length code of like rate and sequence properties.

The structure of variable-length codes required to comply with sequence properties is quite similar to that of fixed-length codes. Various special features, however, arise from the presence of words of different lengths. The requirement of synchronous transmission, coupled with the assumption that each word carries an integer number of information bits, implies that the codeword lengths are integer multiples of a basic word length $n$, where $n$ is the smallest integer for which the bit per symbol ratio $m/n$ is that of two integers. That is, words of length $n$ carry half as many information bits as those of $2n$. Two excellent representatives of variable-length codes, to be discussed in the next case studies, are due to Franaszek.

**Example 3.5:** Choose the same runlength parameters as in Example 3.3, namely $d = 2$ and $k = \infty$. After a tedious process of elimination of sequence states, a code can be constructed as shown in Table 3.12.

<table>
<thead>
<tr>
<th>Data</th>
<th>Code</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>00</td>
</tr>
<tr>
<td>10</td>
<td>0100</td>
</tr>
<tr>
<td>11</td>
<td>1000</td>
</tr>
</tbody>
</table>

If the input symbol is 'zero', '00' would be transmitted. Otherwise, the encoder would transmit a '0100' or a '1000' depending on whether the next symbol was a 'zero' or a 'one', respectively. By inspection it is clear that the three codewords can be cascaded without violating the $d = 2$ channel constraint. The encoding scheme may be readily implemented with a three-state finite-machine encoder. Decoding can be accomplished without explicitly knowing where the blocks of variable
length start or end, that is, the code is self-punctuating (the two-bit synchronization is supposed to be maintained). The code in hand is self-punctuating, because it satisfies the prefix condition. A binary variable-length block code is a set of \( \{c_0, \ldots, c_{M-1}\} \) of \( M \) binary strings. If the codeword \( c_u \) is not the beginning of \( c_v \) for any \( u \neq v \) and for all \( u \), then the code is called a prefix code.

This elementary example illustrates very well the advantage of the variable-length block coding approach and it actually shows how the fixed-length block code with a 128-word dictionary (see Example 3.3, page 55) may be replaced by one with only three words.

\[
\text{Rate 1/2, (2,7) Code}
\]

The variable-length code pointed out in the previous example can be slightly modified to incorporate a maximum runlength constraint. Table 3.13a discloses the code table of the rate 1/2, (2,7) code, which constitutes the bed-rock of the IBM3370 and 3380 high-performance rigid disk files.

<table>
<thead>
<tr>
<th>Data</th>
<th>Code</th>
</tr>
</thead>
<tbody>
<tr>
<td>10</td>
<td>1000</td>
</tr>
<tr>
<td>11</td>
<td>0100</td>
</tr>
<tr>
<td>011</td>
<td>000100</td>
</tr>
<tr>
<td>010</td>
<td>001000</td>
</tr>
<tr>
<td>000</td>
<td>100100</td>
</tr>
<tr>
<td>0011</td>
<td>00100100</td>
</tr>
<tr>
<td>0010</td>
<td>00001000</td>
</tr>
</tbody>
</table>

The encoding of the incoming data is accomplished by dividing the source sequence into two-, three-, and four-bit partitions to match the entries in the code table, and then mapping them into the corresponding channel representations. The next example describes how the codebook is to be used. Let the source sequence be 010111010, then after the appropriate parsing, we obtain

\[
in: 010 11 10 10 ...,\]

which, using Table 3.13a, is transformed into the corresponding output sequence
The companion Table 3.13b shows the same codewords and a permutation of the codeword assignments (there are 24 permutations of the above correspondences). It is worth pointing out here that the assignment rules, which at first sight seem (again) quite arbitrary, are the outcome of a judicious choice, which will become clear in the following.

**TABLE 3.13b - Variable-length synchronous (2,7) code.**

<table>
<thead>
<tr>
<th>Data</th>
<th>Code</th>
</tr>
</thead>
<tbody>
<tr>
<td>10</td>
<td>$\leftarrow \rightarrow$ 0100</td>
</tr>
<tr>
<td>11</td>
<td>$\leftarrow \rightarrow$ 1000</td>
</tr>
<tr>
<td>011</td>
<td>$\leftarrow \rightarrow$ 001000</td>
</tr>
<tr>
<td>010</td>
<td>$\leftarrow \rightarrow$ 100100</td>
</tr>
<tr>
<td>000</td>
<td>$\leftarrow \rightarrow$ 000100</td>
</tr>
<tr>
<td>0011</td>
<td>$\leftarrow \rightarrow$ 00001000</td>
</tr>
<tr>
<td>0010</td>
<td>$\leftarrow \rightarrow$ 00100100</td>
</tr>
</tbody>
</table>

The code satisfies the prefix condition and is therefore self-punctuating. In the case of Table 3.13b, decoding of the received message can be achieved with a shift register of length eight. The incoming message is shifted into the register every two channel clock cycles, the contents of the register are decoded with some logic. Error propagation is limited: any error in a received bit may entail a decoding error in up to two subsequent data bits, the current data bit and up to one preceding data bit. Thus, no error in a received bit is propagated beyond at maximum four decoded data bits. The correspondence table 3.13a as originally presented by Franaszek [5] has the drawback that it needs a shift register of length twelve, which increases error propagation to at most six decoded symbols. This example demonstrates that the allocation of codewords in a variable-length code may have a crucial effect on the error propagation characteristic of the code. How the assignments should be chosen in order to minimize error propagation is up till now an unsolved problem.

Perusal of Table 3.8b, page 57, reveals that the shortest fixed-length block code that generates a (2,7) code has codeword length 34. Evidently, the variable-length synchronous code is much more attractive with respect to hardware requirements.
3.5.7 Look-Ahead Encoding Technique

Another class of design techniques documented in the literature is called *future-dependent* or *look-ahead* (LA) coding. A block code is said to be look-ahead if the encoding and decoding of a current block may depend on upcoming symbols. The coding schemes may also depend on the current state of the channel and on past as well as future symbols. This technique has been used to produce several practical and quite efficient RLL codes. An example of a code design based on the look-ahead method is the rate 2/3 (1,7) code.

*Rate 2/3, (1,7) Code*

Sequences with runlength constraints $d = 1$ and $k = 6$ have the information capacity 0.669, see Table 3.4, page 44, and thus a rate 2/3 code is feasible. A practical encoding and decoding algorithm for such a rate 2/3 code is not published in the literature, but rate 2/3 codes with constraints $(d = 1, k = 7)$ are available in various forms. Jacoby and Kost [6] described a binary two-thirds rate (1,7) code with full-word look-ahead, which is used in a particular magnetic disk file. To understand the algorithm of the 2/3-rate look-ahead code we commence with the basic encoding table, presented in Table 3.14a. The 2/3-rate code is quite similar to a fixed-length block code, where data words of two bits are converted into codewords of three bits. The basic encoding table lists this conversion for the four basic source words. Encoding is done by taking one source word at a time and always looking ahead to the next source word. After conversion of the source symbols to code symbols, and provided there is no violation of the $d$ constraint at the codeword boundaries, the first codeword (the first three bits) will be made final.

<table>
<thead>
<tr>
<th>Data</th>
<th>Code</th>
</tr>
</thead>
<tbody>
<tr>
<td>00</td>
<td>101</td>
</tr>
<tr>
<td>01</td>
<td>100</td>
</tr>
<tr>
<td>10</td>
<td>001</td>
</tr>
<tr>
<td>11</td>
<td>010</td>
</tr>
</tbody>
</table>

There is always the possibility that the last word, up to the point reached in the encoding process, may change when we look ahead to the next word. When the $d$ constraint is violated - there are four combinations of codewords that indeed may lead to this - we require
TABLE 3.14b - Substituting coding table (1,7) code.

<table>
<thead>
<tr>
<th>Data</th>
<th>Code</th>
</tr>
</thead>
<tbody>
<tr>
<td>00.00</td>
<td>101.000</td>
</tr>
<tr>
<td>00.01</td>
<td>100.000</td>
</tr>
<tr>
<td>10.00</td>
<td>001.000</td>
</tr>
<tr>
<td>10.01</td>
<td>010.000</td>
</tr>
</tbody>
</table>

substitutions in order to eliminate successive 'ones'. The process of substitutions in these four combinations is revealed in Table 3.14b.

As can be seen from the examples given in the last part of this chapter, numerous construction techniques have been developed over the last twenty-five years since its inception. A priori, it is difficult to say which of these techniques is 'best'. Variable-length, sliding block, and look-ahead codes can, as we have demonstrated, yield dramatic reduction in complexity of the encoder and decoder for codes of certain rates. Notably codes with rate 1/2 or 2/3 can be designed very efficiently in this manner, and it is scarcely conceivable that one could improve their performance and/or hardware requirements. If the code rate is of the form \( m/n \), \( m \) and \( n \) large, fixed-length block codes may offer a more attractive choice. Look-up tables or enumerative coding may be used; however, in some instances a comprehensive word assignment can be discovered that allows the use of Boolean equations for encoding and decoding, as in the case of EFM and the rate 8/9 (0,3) block code.

3.6 References

Properties of z-constrained Sequences

This chapter extends some of the concepts introduced in Chapter 2 and specializes on sequences with a spectral null at dc. A description is provided of some statistical characteristics of dc-constrained sequences generated by a Markov information source having maximum entropy. A knowledge of ideal, 'maxentropic' sequences with a spectral null at dc is essential to understand the basic trade-offs between the rate of a code and the amount of suppression of low-frequency components.

4.1 Introduction

Binary sequences with spectral nulls at zero frequency have found widespread application in optical and magnetic recording systems. Dc-balanced codes, as they are often called, have a long history and their application is certainly not confined to the recording practice. Since the early days of digital communication over cable, dc-balanced codes have been employed to counter the effects of low-frequency cut-off due to coupling components, isolating transformers, etc. In optical recording, as explained in Chapter 1, dc-balanced codes are employed to circumvent or reduce interaction between the data written on the disc and the servo systems that follow the track. It was also discussed in Chapter 1 that low-frequency disturbances, for example due to fingerprints, may cause completely wrong read-out if the signal falls below the decision level. Errors of this type are avoided by high-pass filtering which is only permissible provided the encoded sequence itself contains no low-frequency components, or, in other words, is dc-balanced.

Common sense tells us that one needs to sacrifice a certain rate loss in order to convert arbitrary data into a dc-balanced sequence. The first question to be addressed in the present chapter is to quantify the maximum rate, that is capacity, given the fact that a sequence is dc-free. The mathematical tools used to derive the capacity of dc-balanced
sequences are in essence a straightforward application of the theory developed in Chapter 2. The results obtained in this chapter allow us to derive a figure of merit of implemented dc-balanced codes that takes into account both the redundancy and the emergent frequency range with suppressed components. The matter of efficiently implementing dc-free codes is considered in Chapter 5, but naturally relies strongly on the basic results developed in this chapter.

4.2 Properties of Dc-balanced Sequences

The running digital sum of a sequence, (in short, RDS) plays a significant role in the analysis and synthesis of codes whose spectrum vanishes at the low-frequency end. Let \( \{ x_i \} = \{ \ldots, x_{-1}, x_0, x_1, \ldots \} \), \( x_i \in \{-1,1\} \) be a binary sequence. The (running) digital sum \( z_i \) is defined as

\[
 z_i = \sum_{j=-\infty}^{i} x_j = z_{i-1} + x_i. \tag{4.1}
\]

\[\begin{align*}
\text{Input} & \quad 0 \quad 0 \quad 1 \quad 0 \quad 0 \quad 0 \quad 1 \quad 0 \quad 1 \\
\text{Write} & \quad \text{signal} \\
\text{Levels} \quad \{x_i\} & \quad -1 \quad -1 \quad -1 \quad +1 \quad +1 \quad +1 \quad -1 \quad -1 \\
\text{RDS} & +3 \\
\frac{1}{j=\infty} x_i & 0 \quad 3
\end{align*}\]

\[\text{FIGURE 4.1 - Running digital sum versus time. Input symbols are translated into the write signal and the channel bits } x_i.\]

Figure 4.1 portrays the various signals. If \( z_i \) is bounded, the spectral density vanishes at dc. The proof is straightforward. The power spectral density function \( H(\omega) \) is defined as
where the expectation operator \( E() \) is the expected value over the ensemble of sequences \( \{x_i\} \). Since by assumption \( z_i \) is bounded we have

\[
|z_i| = \left| \sum_{m=-M}^{M} x_m \right| \leq B, \text{ for some } B < \infty. \tag{4.3}
\]

So,

\[
\frac{1}{2M+1} \left| \sum_{m=-M}^{M} x_m \right|^2 \leq \frac{B^2}{2M+1}
\]

and

\[
H(0) = \lim_{M \to \infty} E \left[ \frac{1}{2M+1} \left| \sum_{m=-M}^{M} x_m \right|^2 \right] = 0.
\]

There is a remarkable and very useful relation between the sum variance \( s^2 = E(z^2) \) and the width of the spectral notch, that is the range of frequencies with suppressed components, of sequences with a spectral null at dc. The variance of the running digital sum may be expressed in terms of the auto-correlation function \( R_x(i) \) of the dc-balanced sequence. To this end, consider

\[ z_m - z_0 = x_1 + x_2 + \cdots + x_m. \]

The variance of the variable \( z_m - z_0 \) is

\[
2s^2 - 2E(z_i z_{i+m}) = \sum_{j=-m+1}^{m-1} (m - |j|)R_x(j)
\]

\[
= m \sum_{j=-m+1}^{m-1} R_x(j) - 2 \sum_{j=1}^{m-1} jR_x(j).
\]

Let \( H_x(\omega) \) denote the power spectral density function of the sequence. Assuming the function \( H_x(\omega) \) is more or less 'well behaved', that is, it is smooth and \( H_x(\omega) \sim A\omega^2 \), \( \omega < < 1 \), we take the limit for \( m \to \infty \) and use
\[
\lim_{m \to \infty} E(z_{i+m}^2) = 0
\]

\[
\lim_{m \to \infty} \sum_{j=-m+1}^{m-1} R_x(j) = H_x(0) = 0,
\]

we get

\[
s^2_x = - \sum_{i=1}^{\infty} i R_x(i).
\]

(4.4)

Let us now, for the sake of convenience, suppose that the auto-correlation function \( R_x(i) \) is an exponentially decaying function of \( i \), or in mathematical terminology

\[
R_x(0) = 1,
\]

\[
R_x(i) = \rho r^{|i|}, \quad i \neq 0, \quad |r| < 1,
\]

(4.5)

where the constant \( \rho \) is chosen in order that the spectrum vanishes at the zero frequency, or \( H_x(0) = 0 \). Thus

\[
H_x'(0) = \sum_{i=-\infty}^{\infty} R_x(i) = 1 + 2 \sum_{i=1}^{\infty} R_x(i) = 0,
\]

(4.6)

whence

\[
\rho = -\frac{1}{2} \frac{(1-r)}{r}. \quad (4.7)
\]

The corresponding power spectral density function, \( H_x(\omega) \), is

\[
H_x'(\omega) = \sum_{i=-\infty}^{\infty} R_x(i) e^{-j\omega i}
\]

\[
= 1 + \rho (re^{-j\omega} + r^2 e^{-j2\omega} + \ldots + re^{j\omega} + r^2 e^{j2\omega} + \ldots).
\]

(4.8)

Now, since

\[
\sum_{k=0}^{\infty} x^k = \frac{1}{1-x}
\]

then
Properties of $z$-constrained Sequences

\[
H_z(w) = \rho \sum_{i=0}^{\infty} (re^{-j\omega i}) + \rho \sum_{i=0}^{\infty} (re^{j\omega i}) - 2\rho + 1
\]

\[
= (1 + r) \frac{1 - \cos \omega}{1 + r^2 - 2r \cos \omega}.
\]

(4.9)

**FIGURE 4.2** - Power spectral density function $H_z(w)$ against frequency $\omega$ of sequences with an exponential decay of its auto-correlation function $R_z(i) = pr^{i|i}$ with the quantity $r$ as a parameter. By way of example, the cut-off frequency is shown for the curve with parameter $r = 0$. It can be noticed that a negative value of the parameter $r$ shifts the power towards the upper end of the baseband.

Figure 4.2 shows the power spectral density function $H_z(w)$ against the frequency $\omega$ with the quantity $r$ as a parameter. The use of (4.4) yields an expression for the sum variance:

\[
\sigma_z^2 = -\sum_{i=1}^{\infty} iR_z(i) = \frac{1}{2(1 - r)}.
\]

(4.10)

Let $H(\omega)$ denote the power density function of a sequence with vanishing power at dc. The width of the spectral notch can be quantified by a parameter called the cut-off frequency. The cut-off frequency $\omega_0$ of a dc-constrained sequence is defined by (see also Figures 4.2 and 4.4):
Specifically, substitution of $H_x(\omega)$ yields

$$H'_x(\omega_0) = (1 + r) \frac{1 - \cos \omega_0}{1 + r^2 - 2r \cos \omega_0} = \frac{1}{2},$$

or

$$1 - \cos \omega_0 = \frac{1}{2} (1 - r)^2.$$

For small values of $1 - r$ we may use the approximation

$$\omega_0 \approx 1 - r.$$  \hspace{1cm} (4.12)

A combination of (4.10) and (4.12) and a little rearrangement yields an approximation to the cut-off frequency $\omega_0$ in terms of the sum variance:

$$2s^2\omega_0 \approx 1.$$ \hspace{1cm} (4.13)

Thus, for the sequence with an exponentially decaying auto-correlation function, we conclude that the cut-off frequency is approximately inversely proportional to the variance of the running digital sum. When the auto-correlation function does not obey the assumed exponential decay, the situation is more complicated, and no general statement can be made. In spite of the fact that the relationship between sum variance and cut-off frequency is only correct for dc-balanced sequences with an exponentially decaying auto-correlation, recent studies have shown that it also applies to sequences generated by implemented channel codes. Detailed computations of samples of implemented channel codes have revealed that the reciprocal relation (4.13) between cut-off frequency and sum variance is very accurate. This result has motivated us to apply the sum variance as a valuable criterion of the low-frequency characteristics of a channel code, which is of practical significance since the sum variance of a dc-free sequence can often be evaluated by simple calculations even though the auto-correlation function and corresponding spectrum are complicated functions.
4.3 Capacity of Dc-constrained Sequences

It will be apparent to the reader, even before matters of implementation are considered, that spectral shaping of a sequence by removing the low-frequency components can be achieved only at the price of a certain rate loss. In order to provide an answer to such a fundamental question, we will study binary sequences \( \{x_i\} \) that assume a finite number of sum values, that is, sequences that meet the following condition:

\[
N_1 \leq z_i \leq N_2,
\]

where \( N_1 \) and \( N_2 \) are two (finite) constants, \( N_2 > N_1 \). Sequences that have a bound to the number of assumed sum values are termed \( z \) (-constrained) or RDS-constrained sequences. The total number of sum values a sequence assumes, denoted by

\[
N = N_2 - N_1 + 1,
\]

(4.14)
is often called the \textit{digital sum variation (DSV)}. A very important question is: How much loss in information rate does one incur by demanding the running digital sum of a sequence to stay within certain limits? Chien addressed the former problem of establishing the information capacity of \( (z) \) sequences as a function of the digital sum variation. In essence, the solution of this problem is provided in Chapter 2. We commence by restating the previous channel constraints in terms of channel states of a finite-state machine.

Taking \( z_i \) at any instant \( i \) as the state of the signal stream \( \{x_i\} \), then the bounds to \( z_i \) define a set of \( N \) allowable states denoted by \( \{\sigma_1, \ldots, \sigma_N\} \). Each transmission of an additional symbol \( x_i \) can be considered as a transition from one state to another. Each transition can be represented by a connection (or adjacency) matrix. For the \( N \)-state source, an \( N \times N \) connection matrix \( D_N \) is defined by \( d_N(i,j) = 1 \) if a transition from state \( \sigma_i \) to state \( \sigma_j \) is allowable and \( d_N(i,j) = 0 \) otherwise. The connection matrix \( D_N \) for the \( z \)-constrained sequences is given by

\[
d_N(i+1,i) = d_N(i,i+1) = 1, \quad i = 1, 2, \ldots, N - 1,
\]

\[
d_N(i,j) = 0, \quad \text{otherwise.} \quad (4.15)
\]

\( D_N \) is a symmetric Toeplitz matrix, and it has ones in the upper- and sub-diagonal and zeros elsewhere. The process discussed above is also known as the random-walk problem with reflecting walls. As an example, we have written down the matrix \( D_N \) for a source that assumes at maximum \( N = 5 \) sum values.
Figure 4.3 portrays the Mealy-type along with the equivalent Moore-type finite-state transition diagram for a sequence that assumes $N = 3$ states. In the Mealy-type finite-state machine the states, embodied by circles, are connected by arrows which represent allowed transitions from one state to the other. Along the edges we have indicated the symbols emitted by the machine when the chain goes from one state to the other. The equivalent Moore-type finite-state machine, see 4.3b, has six states.

The information source model enables us to compute the Shannon capacity of the constrained channel. As was already clarified in Chapter 2, the capacity equals the base-two logarithm of the largest real eigenvalue of the connection matrix $D_N$. The following analysis provides a closed-form expression for the capacity of an RDS-constrained channel.

Let

$$\Phi_N(z) = \det[zI - D_N]$$  \hspace{1cm} (4.16)
designate the characteristic polynomial of $D_N$. The first few polynomials $\Phi_N(z)$ can be evaluated by hand:

\[
\begin{align*}
\Phi_1(z) &= z, \\
\Phi_2(z) &= z^2 - 1, \\
\Phi_3(z) &= z^3 - 2z, \\
\Phi_4(z) &= z^4 - 3z^2 + 1.
\end{align*}
\]

The polynomials $\Phi_N(z)$ have some interesting properties. For $N > 2$ we can write down the following recursion relation:

\[
\Phi_N(z) = z\Phi_{N-1}(z) - \Phi_{N-2}(z), \quad N = 3, 4, \ldots. \tag{4.17}
\]

Equation (4.17) holds for $N = 2$ provided we define $\Phi_0(z) = 1$. To prove (4.17), we expand the determinant in (4.16) with respect to the first column. The eigenvalues of the matrix $D_N$, which are denoted by $\lambda_1, \ldots, \lambda_N$ are the zeros of $\Phi_N(z)$. The associated eigenvectors, denoted by $v_i, i = 1, 2, \ldots, N$, are

\[
v_i = (\Phi_0(\lambda_i), \ \Phi_1(\lambda_i), \ \ldots, \ \Phi_{N-1}(\lambda_i))^T. \tag{4.18}
\]

Introducing the notation $z = 2 \cos x$, (4.17) becomes

\[
\Phi_N(2 \cos x) = (2 \cos x)\Phi_{N-1}(2 \cos x) - \Phi_{N-2}(2 \cos x), \quad N = 2, 3, \ldots.
\]

Treating this recursion relation as a difference equation of $\Phi_N(2 \cos x)$, one can express $\Phi_N(z)$ in an alternative form. The equation $p^2 = (2 \cos x)p - 1$ has roots $e^{i\beta}$ so that

\[
\Phi_N(2 \cos x) = a_1 e^{iNx} + a_2 e^{-iNx},
\]

where the constants $a_1$ and $a_2$ can be determined from the cases $N = 0$ and $N = 1$. Thus

\[
\Phi_N(2 \cos x) = \frac{\sin(N + 1)x}{\sin x},
\]

or

\[
\Phi_N(z) = \frac{\sin[(N + 1)\cos^{-1}(z/2)]}{\sin[\cos^{-1}(z/2)]}. \tag{4.19}
\]

The zeros of $\Phi_N(z)$ are, as can easily be seen in (4.19),
\[
\lambda_i = 2 \cos \frac{in}{N+1}, \quad i = 1, 2, \ldots, N. \tag{4.20}
\]

Thus, the maximum real eigenvalue, i.e., the maximum zero of (4.19), of the matrix defined by (4.15) is given by the following simple expression:

\[
\lambda = \max \{ \lambda_i \} = 2 \cos \frac{\pi}{N+1}, \tag{4.21}
\]

and thus the capacity of the z-constrained channel is

\[
C(N) = \log_2 \lambda = 1 + \log_2 \cos \frac{\pi}{N+1}. \tag{4.22}
\]

Table 4.1 lists the capacity \( C(N) \) and sum variance versus the digital sum variation \( N \), which are deduced using (4.22) and (4.30).

<table>
<thead>
<tr>
<th>( N )</th>
<th>( C(N) )</th>
<th>( \sigma^2_2(N) )</th>
</tr>
</thead>
<tbody>
<tr>
<td>3</td>
<td>0.5000</td>
<td>0.5000</td>
</tr>
<tr>
<td>4</td>
<td>0.6942</td>
<td>0.8028</td>
</tr>
<tr>
<td>5</td>
<td>0.7925</td>
<td>1.1667</td>
</tr>
<tr>
<td>6</td>
<td>0.8495</td>
<td>1.5940</td>
</tr>
<tr>
<td>7</td>
<td>0.8858</td>
<td>2.0858</td>
</tr>
<tr>
<td>8</td>
<td>0.9103</td>
<td>2.6424</td>
</tr>
<tr>
<td>9</td>
<td>0.9276</td>
<td>3.2639</td>
</tr>
<tr>
<td>10</td>
<td>0.9403</td>
<td>3.9506</td>
</tr>
<tr>
<td>11</td>
<td>0.9500</td>
<td>4.7026</td>
</tr>
</tbody>
</table>

The quantity called \textit{sum variance} \( \sigma^2_2(N) \), will be explained in the next section. It can be seen that the results shown in Table 4.1 are intuitively correct for extreme values of \( N \). If \( N \to \infty \) there is more degree of freedom to allow sequences, which, of course, entails a vanishingly small amount of rate loss. Another conclusion to be drawn from Table 4.1 is that the capacity is a strong function of the prescribed digital sum constraint, especially when the digital sum variation \( N \) is small. It can also be seen that the sum constraint is not very expensive in terms of rate loss when \( N \) is relatively large. For instance, a sequence that takes at maximum \( N = 8 \) sum values has a capacity \( C(8) = 0.91 \), which implies a rate loss of less than 10 percent.
4.4 Spectra of Maxentropic Dc-constrained Sequences

In this section we shall proceed with our analysis of RDS-constrained sequences and derive expressions for the power density function. To that end, let the eigenvector associated with the largest eigenvalue of $D_N$ be denoted by $\hat{\phi}$. The state-transition probability matrix $Q$ that maximizes the entropy of the $N$-state source when the connection matrix is given, is (see Chapter 2)

$$Q = \frac{1}{\lambda} A^{-1} D_N A,$$

where $A = \text{diag}(\hat{\phi}, \ldots, \hat{\phi}_N)$. Since $D$ is symmetrical, the stationary probability $\pi_i$ associated with state $\sigma_i$ is given by

$$\pi_i = \rho \hat{\phi}_i^2 = \rho \Phi_{N-1}^2(\lambda), i = 1, 2, \ldots, N,$$

where $\rho$ is chosen to retain the normalizing condition

$$\sum \pi_i = 1.$$

After an evaluation, we obtain the following simple expression for the state distribution:

$$\pi_i = \frac{2}{N+1} \sin^2 \frac{\pi i}{N+1}, i = 1, 2, \ldots, N. \quad (4.23)$$

As usual, a simple example may serve to clarify the method.

**Example 4.1:** Consider a sequence with digital sum variation $N = 5$. The characteristic equation of

$$D_5 = \begin{bmatrix}
0 & 1 & 0 & 0 & 0 \\
1 & 0 & 1 & 0 & 0 \\
0 & 1 & 0 & 1 & 0 \\
0 & 0 & 1 & 0 & 1 \\
0 & 0 & 0 & 1 & 0
\end{bmatrix}$$

is

$$z(z^2 - 1)(z^2 - 3) = 0,$$
so that the maximum real eigenvalue is $\lambda = \sqrt{3}$. The capacity of the constrained sequence is

$$C(5) = \log_2 \lambda = \log_2 \sqrt{3} \approx 0.792,$$

which coincides with (4.22) and Table 4.1. The corresponding eigenvector is

$$\hat{\psi} = (1, \sqrt{3}, 2, \sqrt{3}, 1)^T.$$

The evaluation of the transition probabilities under the condition that the Markov source is maxentropic, is now a matter of a substitution, or

$$Q = \begin{bmatrix}
0 & 1 & 0 & 0 & 0 \\
1/3 & 0 & 2/3 & 0 & 0 \\
0 & 1/2 & 0 & 1/2 & 0 \\
0 & 0 & 2/3 & 0 & 1/3 \\
0 & 0 & 0 & 1 & 0 \\
\end{bmatrix}.$$  

The stationary probability distribution vector is

$$\bar{\pi} = (1/12, 1/4, 1/3, 1/4, 1/12).$$

Connected with the unifilar Markov source is the auto-correlation function of the message $\{z_i\}$ emitted by the source, or

$$R_\psi(k) = E\{z_i \bar{z}_{i+k}\}.$$  

Equivalently, the power spectral density of the emitted sequence $\{z_i\}$ is

$$H_A(\omega) = \sum_{k=-\infty}^{\infty} R_\psi(k) e^{-jk\omega} = R_\psi(0) + 2 \sum_{k=1}^{\infty} R_\psi(k) \cos k\omega. \quad (4.24)$$  

For the sake of convenience, we shall assume $E(z_i) = 0$. The auto-correlation function of the sequence $\{z_i\}$, may be expressed as

$$R_\psi(k) = \bar{\zeta}^T \Pi Q^k \zeta,$$  

where $\bar{\zeta}$ indicates the column vector

$$\bar{\zeta}^T = (\zeta(\sigma_1), \zeta(\sigma_2), \ldots, \zeta(\sigma_N))$$

and $\Pi$ is the diagonal matrix $\Pi = \text{diag}(\pi_1, \ldots, \pi_N)$. Since it is assumed that the values of $\{z_i\}$ are centered around zero, we obtain
\[ \zeta(\sigma_i) = i - \frac{N + 1}{2}, \quad i = 1, 2, \ldots, N. \]

In closed form
\[ H_x(\omega) = -R_z(0) + \zeta^T \Pi (I - e^{j\omega} Q)^{-1} \zeta + \zeta^T \Pi (I - e^{-j\omega} Q)^{-1} \zeta. \quad (4.26) \]

This expression is not very well suited for numerical purposes; the following analysis is more attractive in that respect.

The \( N \times N \) state-transition matrix \( Q \) has \( N \) distinct eigenvalues \( \{1, \mu_2, \ldots, \mu_N\} \) and corresponding left eigenvectors \( u_1 = \pi, u_2, \ldots, u_N \). The eigenvectors are distinct and constitute a basis, so that
\[ \sum_{k=1}^{N} \zeta^T \Pi Q^k \zeta = \left( \eta_1 \bar{\pi} + \sum_{j=2}^{N} \mu_j |k| j \right) \zeta. \quad (4.27) \]

The constants \( \eta_i, i = 1, \ldots, N \) can be found by evaluating \( R_z(k), k = 0, \ldots, N - 1 \). After having established an expression for the auto-correlation function of the sequence \( \{z_i\} \) we must find a relation for the auto-correlation function for the sequence \( \{x_i\} \). By definition we have
\[ z_i = z_{i-1} + x_i. \]

The corresponding auto-correlation function of the \( z \)-constrained sequence \( \{x_i\} \) in terms of the auto-correlation function \( R_z(k) \) is
\[ R_z(k) = \begin{cases} 2R_z(k) - R_z(k - 1) - R_z(k + 1), & k \neq 0, \\ 1, & k = 0, \end{cases} \quad (4.28) \]
or in frequency domain terms
\[ H_z(\omega) = \frac{H_x(\omega)}{2(1 - \cos \omega)}, \quad (4.29) \]

where \( H_x(\omega) \) and \( H_z(\omega) \) denote the spectra of \( \{x_i\} \) and corresponding \( \{z_i\} \), respectively.

By way of illustration, we have plotted in Figure 4.4 the power spectral density function of maxentropic \( z \)-constrained sequences for various values of the digital sum variation \( N \). It can be seen that the power density function indeed vanishes, and is small as well in the neighbourhood of the zero frequency. It can also be observed that the frequency range where the power spectral density function is small, that
is, the width of the spectral notch, becomes smaller when the digital sum variation \( N \) is allowed to increase.

![Graph showing power density function \( H_{\delta}(\omega) \) of maxentropic \( (z) \) sequences against frequency \( \omega \) with digital sum variation \( N \) as a parameter. For the case \( N = 5 \), we have indicated the cut-off frequency \( \omega_0 \).](image)

**FIGURE 4.4** - Power density function \( H_{\delta}(\omega) \) of maxentropic \( (z) \) sequences against frequency \( \omega \) with digital sum variation \( N \) as a parameter. For the case \( N = 5 \), we have indicated the cut-off frequency \( \omega_0 \).

At this juncture we have completed our discussion of dc-balanced sequences. Our next task is to determine some simple bounds to the performance of dc-balanced sequences in terms of the spectral notch width. These bounds underlie our subsequent discussion, in Chapter 5, of the performance of implemented dc-balanced codes. We commence, since it is closely related to the spectral notch width, by computing the sum variance of maxentropic dc-balanced sequences.

The sum variance \( E\{z^2\} \) of a maxentropic \( (z) \) sequence, denoted by \( \sigma^2(N) \), is given by

\[
\sigma^2(N) = E\{z_i^2\} = \sum_{k=1}^{N} \left( \frac{N + 1}{2} - k \right)^2 \pi_k.
\]

Using (4.23) and working out yields
\[
\sigma_z^2(N) = \frac{2}{N+1} \sum_{k=1}^{N} \left( \frac{N + 1}{2} - k \right)^2 \sin^2 \frac{\pi k}{N+1}.
\] (4.30)

Results of computations are collected in Table 4.1.

From Figure 4.4 we may read that in the case \( N = 5 \), the indicated cut-off frequency is approximately 0.43. This given plus the fact that the sum variance \( \sigma^2(5) = 7/6 \), (see Table 4.1) we find that twice the product of sum variance and cut-off frequency is 1.008, which, at least in this specific case, is quite close to the value predicted by (4.13).

A plot of the sum variance versus the redundancy \( 1 - C(N) \), see Figure 4.5, provides more insight; it reveals that the relationship between the logarithms of the sum variance and the redundancy is approximately linear. The immediate engineering implications of the preceding results are quite interesting. The type of curve shown in Figure 4.5 presents the designer with a spectral budget, that is, if the designer desires a certain width of the spectral notch, he knows the price in terms of code redundancy.

![FIGURE 4.5 - Sum variance versus redundancy of maxentropic (z) sequences. The relationship is plotted as a solid line; note, however, that only a discrete set of points is achievable. We may observe that the relationship between the sum variance and the redundancy is approximately linear on the log scales used.](image)

It is worth noting that the graphical results of Figure 4.5 may be expressed analytically for large values of the digital sum variation \( N \). For
asymptotically large digital sum variation $N$, the sum variance and capacity can be approximated by

$$\sigma_2^2(N) \approx \frac{2}{N + 1} \int_{k=0}^{N} \left( \frac{N + 1}{2} - k \right)^2 \sin^2 \frac{\pi k}{N + 1} \, dk$$

$$\approx \left( \frac{1}{12} - \frac{1}{2\pi^2} \right)(N + 1)^2, \quad N \gg 1 \tag{4.31}$$

and in a similar way

$$C(N) \approx 1 - \frac{\pi^2}{2\ln 2} \frac{1}{(N + 1)^2}, \quad N \gg 1. \tag{4.32}$$

The former approximations plus perusal of (4.22) and (4.23) lead to a fundamental relation between the redundancy $1 - C(N)$ and the sum variance of a maxentropic ($z$) sequence, namely

$$0.25 \geq (1 - C(N))\sigma_2^2(N) > \frac{\pi^2/6 - 1}{4\ln 2} = 0.2326. \tag{4.33}$$

Actually, the right-hand bound is within 1% accuracy for $N > 9$. The preceding relationship between sum variance (and implicitly according to (4.13) the cut-off frequency) and redundancy is very interesting, since it reflects what we intuitively expect, and it illustrates very well the phrase that there is no such thing as a free lunch. For a wider frequency range of suppressed components one has to pay more in terms of redundancy of the sequence. Relation (4.33) shall be used in the subsequent chapter to establish a general figure of merit of implemented dc-constrained codes.

In summarizing the lessons to be learnt from the theory, it should be borne in mind that the treatment given here applies solely to RDS constrained sequences. If the sequences comply with other constraints as well, for example runlength constraints, then, in general, the results derived above lose their validity.
Performance of Dc-balanced Codes

Many of the most important block codes for spectral shaping purposes fall into the category of dc-balanced codes. In digital transmission it is sometimes desirable for the channel stream to have low power near zero frequency. Suppression of the low-frequency components is usually achieved by restricting the unbalance of the transmitted positive and negative pulses. Rate and spectral properties of unbalance constrained codes with binary symbols based on simple bi-mode coding schemes are calculated.

5.1 Introduction

Whereas the previous chapter provides a theoretical basis of the structural properties of dc-balanced sequences, this chapter turns to the issue of a more practical nature of how to efficiently implement encoder algorithms that generate dc-balanced sequences.

Practical coding schemes devised to achieve suppression of low-frequency components are mostly constituted by block codes. The source digits are grouped in source words of $m$ digits; the source words are translated using a conversion table known as a codebook into blocks of $n$ digits. The essential principle of operation of a channel encoder that translates arbitrary source data into a dc-free channel sequence is remarkably simple. The approaches which have actually been used for dc-balanced code design are basically three in number:

- zero-disparity code,
- low-disparity code,
- polarity bit code.

The disparity of a codeword is defined as the excess of the number of 'ones' over the number of 'zeros' in the codeword; thus the codewords 000110 and 100111 have disparity -2 and +2, respectively. An important special case are zero-disparity codewords, which contain equal numbers of 'ones' and 'zeros'. The obvious method for the construction of
dc-balanced codes is to employ codewords that contain an equal number of 'ones' and 'zeros', or stated alternatively, to employ zero-disparity codewords which have a one-to-one correspondence with the source words.

A logical step, then, is to extend this mechanism to the low-disparity code, where the translations are not one-to-one. The source words operate with two alternative translations (or modes) being of equal or opposite disparity, any of which is interpreted by the decoder in the same way. The zero-disparity words are uniquely allocated to the source words. Other codewords are allocated in pairs of equal and opposite disparity. During transmission, the choice of a specific translation is made in such a way that the accumulated disparity, or the running digital sum, of the encoded sequence, after transmission of the new codeword, is as close to zero as possible. The running digital sum (RDS) is defined for a binary stream as the accumulated sum of 'ones' and 'zeros' (a zero counted as -1) counted from the start of the transmission. Both of the basic approaches to dc-balanced coding are due to Cattermole [1].

The third coding method to be discussed was devised by Bowers [2] who proposed a slightly different construction of dc-balanced codes as being attractive because no look-up tables are required for encoding and decoding. They proposed a code with \((n-1)\) source symbols that are mapped without modification onto the first \((n-1)\) symbols of the codeword. The additional \(n\)th symbol of the codeword, called the polarity bit, is used to identify the polarity of the transmitted codeword.

The outline of this chapter is as follows. In order to get some insight into the efficiency of the afore-mentioned construction techniques we shall evaluate the spectral properties of their respective code streams. Of course, the performance may be evaluated for any given code structure by resorting to numerical computation. The theory provided in [3] furnishes efficient procedures for the computation of the power spectral density function of block coded signals produced by an encoder that can be modelled by a finite-state machine. Thus the prospect of numerical analysis need not be depressing. The difficulty is rather that we prefer to know the dependence of the code's performance upon the various design parameters, and such dependencies are difficult to assess by numerical analysis. Fortunately, the structure of bi-mode dc-balanced codes allows us to derive simple expressions for the rate and power spectral density functions. In Section 5.2, the power density function of low-disparity based channel codes is established. The variance of the running digital
sum (in short, sum variance) of the channel codes, adopted here as a criterion of the suppression of the energy near dc, is calculated in Section 5.2.

The theoretical development of Chapter 4 demonstrates there is a relationship between the performance of the code expressed in the notch width and the redundancy of the code. Coding efficiency and ease of implementation are to a large extent incompatible, and we are, for the moment, solely concerned with their mathematical properties. One question of significance is that of code efficiency expressed in terms of spectral notch width and redundancy. The performance of maxentropic dc-balanced sequences, whose properties are studied in Chapter 4, sets a standard by which the performance of dc-balanced code implementations may be measured. In Section 5.3, using the theory developed in Chapter 4, we intend to answer the question: How good are the implemented channel codes for the given redundancy and concomitant suppression of low-frequency components? Engineering examples are discussed in Section 5.4.

5.2 Simple Coding Schemes

5.2.1 Zero-Disparity Coding Schemes

First, some properties of coding schemes constituted by codewords with an equal number of positive and negative pulses, usually termed zero-disparity codewords, are discussed. Clearly, zero-disparity codewords are possible only if the codeword length $n$ is even. The number $N_0$ of binary channel symbols ($n$ even) is given by the binomial coefficient

$$N_0 = \binom{n}{n/2}.$$

Table 5.1 shows the number of zero-disparity codewords as a function of the codeword length $n$. Table 5.1 also presents the code rate $R$, which is defined by

$$R = \frac{1}{n} \log_2 N_0.$$  \hspace{1cm} (5.1)

The zero-disparity codewords are concatenated without a merging rule.

Practical coding arrangements demand that the number of codewords is a power of two, so that a subset of the $N_0$ available codewords has to be used, which as a result effectively lowers the code
TABLE 5.1 - Number of zero-disparity codewords and code rate versus codeword length \( n \).

<table>
<thead>
<tr>
<th>( n )</th>
<th>( N_0 )</th>
<th>( R )</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>2</td>
<td>0.500</td>
</tr>
<tr>
<td>4</td>
<td>6</td>
<td>0.646</td>
</tr>
<tr>
<td>6</td>
<td>20</td>
<td>0.720</td>
</tr>
<tr>
<td>8</td>
<td>70</td>
<td>0.766</td>
</tr>
<tr>
<td>10</td>
<td>252</td>
<td>0.798</td>
</tr>
<tr>
<td>12</td>
<td>924</td>
<td>0.821</td>
</tr>
<tr>
<td>14</td>
<td>3432</td>
<td>0.839</td>
</tr>
<tr>
<td>16</td>
<td>12870</td>
<td>0.853</td>
</tr>
<tr>
<td>18</td>
<td>48620</td>
<td>0.865</td>
</tr>
<tr>
<td>20</td>
<td>184756</td>
<td>0.875</td>
</tr>
</tbody>
</table>

Rate \( R \). For the sake of mathematical convenience, only 'full set' coding schemes are considered here.

### 5.2.2 Low-Disparity Coding Schemes

Up to this point, we have been concerned with the category of dc-balanced codes which solely makes use of codewords of zero-disparity. A generalization of the former coding principle leads, in a quite straightforward manner, to the alternate or low-disparity coding technique. Besides the set of zero-disparity codewords, sets of codewords with non-zero disparity are used. The archetypical code has two alternate representations of the source words. The two alternate representations have opposite disparity; the choice of positive or negative representation is stipulated by the polarity of the running digital sum just before transmission of the new codeword. The encoder opts for a particular channel representation with the aim to minimize the absolute value of the running digital sum after transmission of the new codeword. Zero-disparity codewords can in principle be used in both modes, and are usually uniquely allocated to source words.

The assignment of the source words to the various codewords is to some extent arbitrary. There is, however, one important requirement that has to be taken into account: the decoding should be state-independent to circumvent serious error propagation.

Obviously, if more subsets of codewords are used, the number of codewords is larger than in the case of zero-disparity encoding (assuming
equal codeword length). Consequently, this allows a larger maximum code rate for a given codeword length. Unfortunately, as we shall see in a moment, the power in the low-frequency range will also increase if more subsets are used, so that a trade-off between code rate and low-frequency content has to be found. In the following, some basic properties of low-disparity coding are derived.

Let a codeword of length \( n \), \( n \) even, consist of binary symbols \( x_i, 1 \leq i \leq n, x_i \in \{-1, 1\} \). The disparity \( d \) of a codeword is defined by

\[
d = \sum_{i=1}^{n} x_i.
\]  

The codebook comprises two sets, or pages, denoted by \( S_+ \) and \( S_- \), respectively. The set \( S_+ \) comprises codewords of zero and positive disparity, and the codewords in set \( S_- \) have zero and negative disparity. Set \( S_+ \) comprises \( K+1 \) subsets, designated by \( S_0, S_1, S_2, \ldots, S_K \), \((K \leq n/2)\). The elements of the subsets \( S_j \) are all possible codewords with disparity \( 2^j, 0 \leq j \leq K \). The codewords in \( S_+ \) are found by inversion of all \( n \) symbols of the codewords in set \( S_- \) and vice versa. The cardinality \( N_j \) of the subset \( S_j \) is the binomial coefficient

\[
N_j = \binom{n}{n/2 + j}, \quad 0 \leq j \leq K.
\]  

The total available number of codewords in \( S_+ \) or \( S_- \), denoted by \( M \), is

\[
M = |S_+| = |S_-| = \sum_{j=0}^{K} N_j.
\]

The code rate is simply given by

\[
R = \frac{1}{n} \log_2 M.
\]  

An essential part of the low-disparity encoder is a counter which registers the running digital sum. As the disparity of the codewords is selected such that the absolute value of the running digital sum after transmission of the new codeword is minimized, it is not difficult to see that during transmission the running digital sum (RDS) takes on a finite number of values. Without loss of generality it can be assumed (by properly choosing the initial sum value at the beginning of the transmission) that the sum values are symmetrically centered around
zero. The set of values (states) the RDS assumes at the end (or start) of a codeword, termed the terminal or principal states, is a subset of the RDS values the sequence can take.

Let the terminal digital sum after transmission of the $t$th transmitted codeword be $D(t)$. The sum after transmission of the $(t+1)$th codeword is

$$D(t+1) = D(t) \pm d(t+1),$$

where $d(t+1)$ is the disparity of the $(t+1)$th codeword. The sign of the disparity, if $d(t+1) \neq 0$, of the codeword is chosen to minimize the accumulated sum $D(t+1)$. A code with this property is said to be balanced. After a little thought we conclude that the quantity $D(t)$ may assume one of the $2K$ values \{±1, ±3, ..., ±$(2K-1)$\}, provided the encoder is properly initialized. Apparently, when the encoder is ruled by the previous algorithm there exists a bound to the maximum accumulated digital sum. It can easily be verified that the total number of RDS values that the sequence can take, i.e., the digital sum variation, is

$$N = 2(2K - 1 + \frac{n}{2}) + 1 = 4K + n - 1. \quad (5.5)$$

**FIGURE 5.1** - Unbalance trellis diagram. The lines display the running digital sum of each permitted codeword, and so give a visual picture of the encoding process. The thick curve shows the path taken by the codeword ‘+ --- +’ assuming that it emanates from state $s_3$. The diagram is similar to Figure 4.1, page 70, now the time axis is initialized at the start of each codeword.
As an illustration the RDS as a function of symbol time interval is shown in Figure 5.1. The lines display the running digital sum of each of the codewords, and so give a visual picture of the encoding process. A chart like this is called a *trellis diagram*. The code has codeword length $n = 6$ and it uses the maximum number $K + 1 = n/2 + 1 = 4$ subsets. Note the $2K = 6$ allowed sum values at the end of each codeword and also the $N = 4K + n - 1 = 17$ values that the running digital sum can take within a codeword.

The encoder which generates a low-disparity sequence can be modelled as a finite-state machine. The conjunction of a source word and the terminal sum value $D^{(i)}$ determines the actual transmitted codeword and the next terminal sum value $D^{(i+1)}$. Thus, the set of encoder states, (or principal states) designated by $\Sigma = \{\sigma_1, \ldots, \sigma_{2K}\}$, is the set of terminal sum values.

In the computation of the power density function and the sum variance of the encoded stream, we need to know the stationary probability of being in a certain terminal state. For this purpose, assume the source words to be generated by a random independent process, then the signal process $D^{(i)}$ is a simple stationary Markov process. The value that $D^{(i)}$ can take is now related to one of the $2K$ states of the Markov process. The state transition probability matrix $Q$ whose entries are represented by $q_{ij}$, where $q_{ij}$ is the probability that the next codeword will take it to terminal state $\sigma_j$ given that the encoder is currently in state $\sigma_i$, can easily be found.

As an illustration we have written down the matrix $Q$ for $2K = 6$ terminal states:

$$Q = \begin{bmatrix}
p_0 & p_1 & p_2 & p_3 & 0 & 0 \\
0 & p_0 & p_1 & p_2 & p_3 & 0 \\
0 & 0 & p_0 & p_1 & p_2 & p_3 \\
p_3 & p_2 & p_1 & p_0 & 0 & 0 \\
0 & p_3 & p_2 & p_1 & p_0 & 0 \\
0 & 0 & p_3 & p_2 & p_1 & p_0
\end{bmatrix}$$

The transition probability $q_{ij}$ is the proportion of codewords in the mode used in state $\sigma_i$ having the appropriate disparity $d$ for the transition to encoder state $\sigma_j = \sigma_i + d_2$. The transition probability $p_i$ equals the relative number of codewords in subset $S_i$, or $p_i = N_i/M$.

The special structure of the matrix $Q$ allows us to establish a closed expression of the steady state probability vector $\pi$ whose entries are
(\pi_K, ..., \pi_1, \pi_1, ..., \pi_K), where \pi_i is the probability of being in the encoder state \sigma_i with corresponding sum value (2i - 1).

\[
p \pi_{K-i} = \sum_{j=K-i}^{K} p_j, \quad 0 \leq i \leq K - 1,
\]

where \rho is determined from the normalization

\[
\sum_{i=1}^{K} \pi_i = \frac{1}{2}.
\]

With (5.6):

\[
\rho = 2 \sum_{i=1}^{K} ip_i.
\]

The proof of (5.6) is by direct verification of the identity

\[
\pi Q = \pi.
\]

This amounts to the following system of linear equations:

\[
\pi K p_j + \pi_{K-1} p_{j-1} + \pi_1 p_{K-j} + \pi_2 p_{K-j+1} + \ldots = \pi_{K-j}, j = 0, \ldots, K - 1.
\]

After evaluating we arrive at (5.6).

In the next example we give some numerical results.

**Example 5.1:** We have written down in Table 5.2 the complete code specification of a 4-state, low-disparity code with parameters \( n = 4, \ K = 2 \), in terms of the output and state-transition functions. The characters '+' and '-' are used to represent a symbol with value 1 and -1, respectively. The four encoder states are designated by \( \sigma_i, i = 1, \ldots, 4 \).

We notice that there are 11 codewords, so the code rate is \( R = 1/4 \log_2 11 \approx 0.865 \). With the codebook listed in Table 5.2 we can now write down the 4 \times 4 state transition probability matrix:

\[
Q = \frac{1}{11} \begin{bmatrix}
6 & 4 & 1 & 0 \\
0 & 6 & 4 & 1 \\
1 & 4 & 6 & 0 \\
0 & 1 & 4 & 6
\end{bmatrix}.
\]
TABLE 5.2 - Codebook of low-disparity code.
Codeword length $n = 4$, number of codeword subsets $K + 1 = 3$.

<table>
<thead>
<tr>
<th>Source</th>
<th>$\sigma_1$</th>
<th>$\sigma_2$</th>
<th>$\sigma_3$</th>
<th>$\sigma_4$</th>
</tr>
</thead>
<tbody>
<tr>
<td>0000</td>
<td>$-+--$</td>
<td>$--+-$</td>
<td>$-+-+$</td>
<td>$++-+$</td>
</tr>
<tr>
<td>0001</td>
<td>$-++-$</td>
<td>$-+-+$</td>
<td>$-++-$</td>
<td>$--+$</td>
</tr>
<tr>
<td>0010</td>
<td>$-++-+$</td>
<td>$-+--$</td>
<td>$-+-+$</td>
<td>$-++-$</td>
</tr>
<tr>
<td>0011</td>
<td>$++-+$</td>
<td>$+-++$</td>
<td>$-+-+$</td>
<td>$-++-$</td>
</tr>
<tr>
<td>0100</td>
<td>$++-+$</td>
<td>$+-++$</td>
<td>$-+-+$</td>
<td>$-++-$</td>
</tr>
<tr>
<td>0101</td>
<td>$++-+$</td>
<td>$+-++$</td>
<td>$-+-+$</td>
<td>$-++-$</td>
</tr>
<tr>
<td>0110</td>
<td>$++-+$</td>
<td>$+-++$</td>
<td>$-+-+$</td>
<td>$-++-$</td>
</tr>
<tr>
<td>0111</td>
<td>$++-+$</td>
<td>$+-++$</td>
<td>$-+-+$</td>
<td>$-++-$</td>
</tr>
<tr>
<td>1000</td>
<td>$++-+$</td>
<td>$+-++$</td>
<td>$-+-+$</td>
<td>$-++-$</td>
</tr>
<tr>
<td>1001</td>
<td>$++-+$</td>
<td>$+-++$</td>
<td>$-+-+$</td>
<td>$-++-$</td>
</tr>
<tr>
<td>1010</td>
<td>$++-+$</td>
<td>$+-++$</td>
<td>$-+-+$</td>
<td>$-++-$</td>
</tr>
</tbody>
</table>

Using (5.6), we find the steady state probabilities

$$\pi(\sigma_1) = \pi(\sigma_4) = \pi_2 = \frac{1}{12}$$
$$\pi(\sigma_2) = \pi(\sigma_3) = \pi_1 = \frac{5}{12}.$$ 

5.2.3 Computation of the Sum Variance

A notable frequency-domain characteristic of dc-balanced codes, the cut-off frequency, can be estimated by evaluating the sum variance of the code stream. The key to this approach is Justesen's relation (4.13), page 74,

$$2s_2^2\omega_0 \approx 1,$$

which provides a simple relationship between the sum variance $s_2^2$ and the cut-off frequency $\omega_0$ of a dc-balanced sequence. Of course, the reciprocal of the sum variance provides only an approximation to the cut-off frequency. Although this weakness in our result is certainly undesirable, it appears to be a necessary compromise if we are to obtain simple analytical results. The only alternative now available for exactly
computing the cut-off frequency is numerical analysis. The accuracy of Justesen's relation is a topic to be addressed in a later section.

Our objective, in this section, is to derive a simple closed-form expression of the sum variance of dc-balanced, bi-mode channel codes. The process of encoding constituted by the alternate code principle is cyclo-stationary with period $n$ so that the sum variance of the sequence has to be established by averaging the running sum variance over all $n$ symbol positions within the codeword. To this end, we commence by computing the running sum variance at all symbol positions within the codeword.

Let the value of the running digital sum at the $k$th position in a codeword be designated by $z_k$, and assume also that a codeword, denoted by $x(i) = (x(1), \ldots, x(k))$ starts with an initial RDS, denoted by $z_0$, one of the $K$ positive terminal sum values (the statistics of the codewords starting at a state associated with a negative sum value can be found by symmetry). The running digital sum at the $k$th position in the codeword $x(i)$ is

$$z_k = z_0 + \sum_{m=1}^{k} x_m, 1 \leq k \leq n.$$ 

The running sum variance at the $k$th position given $z_0$ is

$$E\{z_k^2 | z_0\} = E \left[\left( z_0 + \sum_{m=1}^{k} x_m \right)^2 \right]$$

$$= E \left[ z_0^2 + \sum_{m=1}^{k} (x_m)^2 + 2z_0 \sum_{m=1}^{k} x_m + 2 \sum_{j_1=1}^{k-1} \sum_{j_2=j_1+1}^{k} x_{j_1} x_{j_2} \right],$$

where the operator $E\{\}$ averages over all codewords $x(i)$ that start with an initial RDS $z_0$. A nice and quite useful property of full codeword subsets is that the expectations $E\{x(i) x(i)\}$ and $E\{x_{j_1} x_{j_2}\}, j_1 \neq j_2$, are not a function of the symbol positions $j_1$ and $j_2$. For clerical convenience we define the short-hand notation: $E\{x_{j_1}^2\} = \mu$ and $E\{x_{j_1} x_{j_2}\} = r_0, 1 \leq j_1, j_2 \leq n, j_1 \neq j_2$. Substitution yields the running sum variance at the $k$th symbol position.
Performance of DC-balanced Codes

\[ E(z^2_k | z_0) = z_0^2 + k + 2k\mu z_0 + k(k - 1)r_0. \]  
(5.8)

The sum variance of an allowed sequence that starts with initial RDS \( z_0 \), designated by \( s_k^2 | z_0 \), is found by averaging the running sum variance over all \( n \) symbol positions of the codeword, or

\[ s_k^2 | z_0 = \frac{1}{n} \sum_{k=1}^{n} E(z^2_k | z_0) \]
\[ = z_0^2 + \frac{n + 1}{2} + \mu(n + 1)z_0 + \frac{1}{3} (n^2 - 1)r_0. \]

The probability that a codeword starts with initial RDS \( z_0 = 2i - 1 \), \( 1 \leq i \leq K \), is \( \pi_i \) so that taking into account the probability that a codeword starts with \( z_0 \) and averaging over the \( 2K \) initial states yields the following expression for the sum variance \( s_k^2 \):

\[ s_k^2 = E(z^2_0) + \frac{n + 1}{2} + \frac{1}{3} (n^2 - 1)r_0 + 2(n + 1)\mu \sum_{i=1}^{K} (2i - 1)\pi_i. \]  
(5.9)

The variance of the terminal sum values, \( E(z^2_t) \), is

\[ E(z^2_0) = 2 \sum_{i=1}^{K} (2i - 1)^2 \pi_i, \]  
(5.10)

The quantity \( \mu \) can be eliminated by noting the periodicity, i.e., \( E(z^2_t) = E(z^2_t) \). Evaluating (5.8) yields

\[ E(z^2_t | z_0) = z_0^2 + n + 2\mu z_0 + n(n - 1)r_0, \]

and after averaging, where the probability of starting with an initial RDS \( z_0 \) is taken into account, we obtain:

\[ E(z^2_t) = E(z^2_0) + n + n(n - 1)r_0 + 4n\mu \sum_{i=1}^{K} (2i - 1)\pi_i, \]

so that with \( E(z^2_t) = E(z^2_t) \) we find

\[ 2\mu \sum_{i=1}^{K} (2i - 1)\pi_i = - \frac{1}{2} (1 + (n - 1)r_0). \]

Substitution in (5.9) yields:
\[ s_k^2 = E(x_0^2) - \frac{1}{6} (n^2 - 1) r_0. \] (5.11)

### 5.2.4 Computation of the Correlation

We now calculate the correlation \( r_0 = E(x_j x_b) \) of the symbols at the \( j \)th and \( j_2 \)th symbol position within the same codeword. It is obvious that \( E(x_j x_b) = 1 \). If \( j_1 \neq j_2 \), some more work is needed. In that case

\[
E(x_{j_1} x_{j_2}) = Pr(x_{j_1} = x_{j_2}) - Pr(x_{j_1} \neq x_{j_2})
\]

\[
= 1 - 2Pr(x_{j_1} \neq x_{j_2}), \; j_1 \neq j_2.
\] (5.12)

Suppose a codeword to be an element of subset \( S_i \subset S_n \). The probability that a symbol at position \( j_1 \) in the codeword equals 1 is

\[
Pr(x_{j_1} = 1 \mid S = S_i) = \frac{1}{n} (\frac{n}{2} + i).
\]

The probability that another symbol at position \( j_2 \neq j_1 \) within the codeword equals -1 is

\[
Pr(x_{j_2} = -1 \mid x_{j_1} = 1, S = S_i) = \frac{2}{n - 1}.
\]

So that

\[
Pr(x_{j_1} \neq x_{j_2} \mid S = S_i) = \frac{n^2 - 4i^2}{2n(n - 1)}.
\]

and using (5.12) yields

\[
E(x_{j_1} x_{j_2} \mid S = S_i) = -\frac{1}{n - 1} \left( 1 - \frac{4}{n} i^2 \right).
\]

If we now take into account the probability \( p_i \) that a codeword is an element of subset \( S_i \subset S_n \), we find for the correlation

\[
r_0 = E(x_j x_b) = \frac{-1}{n - 1} \left\{ 1 - \frac{4}{n} \sum_{i=1}^{K} i^2 p_i \right\}.
\] (5.13)

Combining with (5.11) yields
\[ s_K^2 = 2 \sum_{i=1}^{K} (2i-1)^2 \pi_i + \frac{n+1}{6} \left\{ 1 - \frac{4}{n} \sum_{i=1}^{K} i^2 p_i \right\}. \quad (5.14) \]

Define

\[ u_m = \sum_{i=1}^{K} i^m p_i, \quad m \in \{1,2,3\}. \quad (5.15) \]

A manipulation combining (5.6), (5.7), (5.10), (5.14) and (5.15) yields the variance of the terminal sum values

\[ E[z_i^2] = 2 \sum_{i=1}^{K} (2i-1)^2 \pi_i = \frac{4}{3} \frac{u_3}{u_1} - \frac{1}{3} \quad (5.16) \]

and the variance of the complete sequence

\[ s_K^2 = \frac{4}{3} \frac{u_3}{u_1} + \frac{n-1}{6} - 2 \frac{n+1}{3n} u_2. \quad (5.17) \]

We have now completed our analysis of the classical dc-balanced codes. The analysis has not been particularly difficult because the detailed structure of the encoder was irrelevant, the only requirements can be summarized as follows. Eq. (5.16) solely rests on the assumption of the dc-balanced, bi-mode structure of the transition matrix \( Q \). In (5.17) we further assumed that the expectations are invariant with respect to the location in a codeword, which is true if full codeword sets are employed. The analysis of encoders that do not fulfil these conditions is considerably more involved than that just presented because the detailed structure is of paramount concern. In this case the performance has to be evaluated for any given code structure by resorting to numerical computational procedures. In the following examples, we describe easily calculable cases.

**Example 5.2:** Let \( K = 0 \), then we obtain from (5.17), since \( E[z_i] = 0 \), the following simple expression for the sum variance:
Example 5.3: Another simple result can be obtained, if only two subsets, namely $S_0$ and $S_1$, are used for encoding. Clearly,

$$p_1 = \frac{n}{2(n+1)}.$$ 

After substitution and working out (5.15) and (5.17), we obtain

$$u_1 = u_2 = u_3 = p_1,$$

so that

$$s_1^2 = \frac{4}{3} + \frac{n-1}{6} - 2\frac{n+1}{3n} p_1 = \frac{n+5}{6}.$$ 

Example 5.4: Bowers [2] and Carter [41] advocated a construction of dc-balanced codes where the $(n - 1)$ source symbols are mapped without modification onto the first $(n - 1)$ symbols of the codeword. The additional $n$th symbol of the codeword, called the polarity bit, is used to identify the polarity of the transmitted codeword. The encoder is designed in such a way that the first $(n - 1)$ symbols equal the source symbols and the $n$th symbol, the polarity bit, is set to one. If the digital sum at the start of the transmission of a new codeword and the disparity of the new codeword have the same sign, then all symbols in the codeword (including the polarity bit) are inverted (complemented) before transmission. The accumulated disparity is registered by a reversible counter connected to the output line. If the disparity of the codeword is zero, the polarity of the codeword is randomly chosen. The receiving translator is simpler, since it merely observes a possible inversion of a codeword by inspecting the sign of the polarity bit and, according as this is positive or negative reads the remaining bits directly or complemented.

The analysis developed in the preceding sections for the $K+1$ subsets based channel code allows us to derive an expression for the sum variance of the polarity bit encoding construction. The code rate of the polarity bit code is
\( R = 1 - \frac{1}{n} \).

The number of subsets is \( K + 1 = n/2 + 1 \) (\( n \) even), so that the number of terminal sum values is \( 2K = n \). The effective number of zero-disparity codewords \( N_0 \) is halved by the random choice of the 'polarity' of these words with respect to the maximum number used in the low-disparity coding principle, or

\[ N_0 = \frac{1}{2} \binom{n/2}{n/2}. \]

The number of codewords of non-zero disparity is not changed, or

\[ N_i = \binom{n}{n/2 + i}, \quad 1 \leq i \leq \frac{n}{2}. \]

The total number of codewords of non-negative disparity is

\[ M = N_0 + \sum_{i=1}^{n/2} N_i = 2^{n-1}. \]

Using some properties of binomial coefficients a routine computation yields

\[ u_1 = n \binom{n}{n/2} 2^{-(n+1)}, \]

\[ u_2 = \frac{1}{4} n \]

and

\[ u_3 = \frac{n}{2} u_1. \]

Evaluation of (5.17) yields

\[ s_p^2 = \frac{2n - 1}{3}, \quad (5.18) \]

where \( s_p \) designates the sum variance of the polarity bit encoded sequence.

**Example 5.5:** If all codewords are used, i.e., \( K = n/2 \), then
\[ M = 2^{n-1} + \frac{1}{2} \binom{n}{n/2} \]

and

\[ s_{n/2}^2 = \frac{5n - 1}{6} - \frac{n + 1}{12M} 2^n. \tag{5.19} \]

For other values of the number of subsets used in the encoding table, it was not possible to obtain such simple analytical expressions; however, even with a small computer, the variance \( s_k \) can be calculated as a function of \( K \) and \( n \). The results of the computations are collected in Table 5.3, where the redundancy \( 1 - R \) and the digital sum variation \( N \) of the code are also given (see eq. (5.5)). After a study of Tables 4.1, page 78 and 5.3, several interesting facts now appear.

<table>
<thead>
<tr>
<th>( n )</th>
<th>( K )</th>
<th>( N )</th>
<th>( s_{n/2}^2 )</th>
<th>( 1 - R )</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>0</td>
<td>3</td>
<td>0.50</td>
<td>.050</td>
</tr>
<tr>
<td>2</td>
<td>1</td>
<td>5</td>
<td>1.17</td>
<td>.208</td>
</tr>
<tr>
<td>4</td>
<td>1</td>
<td>7</td>
<td>1.50</td>
<td>.170</td>
</tr>
<tr>
<td>4</td>
<td>2</td>
<td>11</td>
<td>2.56</td>
<td>.135</td>
</tr>
<tr>
<td>6</td>
<td>1</td>
<td>9</td>
<td>1.83</td>
<td>.145</td>
</tr>
<tr>
<td>6</td>
<td>2</td>
<td>13</td>
<td>3.20</td>
<td>.107</td>
</tr>
<tr>
<td>6</td>
<td>3</td>
<td>17</td>
<td>3.94</td>
<td>.101</td>
</tr>
<tr>
<td>8</td>
<td>1</td>
<td>11</td>
<td>2.17</td>
<td>.128</td>
</tr>
<tr>
<td>8</td>
<td>2</td>
<td>15</td>
<td>3.68</td>
<td>.092</td>
</tr>
<tr>
<td>8</td>
<td>3</td>
<td>19</td>
<td>4.92</td>
<td>.083</td>
</tr>
<tr>
<td>8</td>
<td>4</td>
<td>23</td>
<td>5.32</td>
<td>.081</td>
</tr>
</tbody>
</table>

The remarkably simple block code with parameters \( n = 2 \) and \( K = 0 \), termed the bi-phase code, attains 100 percent of the capacity and the sum variance of the maxentropic sequence with digital sum variation \( N = 3 \). The two-state alternate block code with parameters \( n = 2 \) and \( K = 1 \) achieves 100 percent of the capacity and the sum variance of the maxentropic sequence with \( N = 5 \). Other implemented codes that attain the performance of maxentropic dc-balanced sequences were not discovered (probably, they are impossible to construct).

Figure 5.2 shows for several codes the sum variance as a function of the redundancy \( 1 - R \) with \( K \) and \( n \) as parameters. As a reference, the
sum variance is plotted versus the redundancy $1 - C(N)$ of maxentropic $(z)$ sequences (see (4.22) and (4.30)). Notice in the figure that the performance of zero-disparity encoding diverges with growing codeword length from the maxentropic bound. It is also clear from the figure that the use of two codeword subsets, that is $K = 1$, is worth-while in terms of sum variance and redundancy in a large range of the code redundancy.

For the sake of convenience we have opted to compute the sum variance in stead of the cut-off frequency. The justification of this approach is based on relation (4.13), page 74. To validate our course of action, the cut-off frequency was calculated using numerical procedures, and compared with the reciprocal of the sum variance of the coded sequence. Consideration of computational results, in the range of code parameters listed in Table 5.3, reveals that the relation between sum variance and actual cut-off frequency is accurate within a few percent, which is indeed a fascinating result.
5.3 Efficiency of Simple Alternate Codes

We now employ the theory on maxentropic \((z)\) sequences, developed in Chapter 4, to appraise the efficiency of the previously discussed dc-balanced codes. It is customary to define the rate efficiency of an implemented channel code as the ratio of the code rate and the noiseless channel capacity given the channel constraints, or

\[
\eta = \frac{R}{C(N)},
\]

where \(\eta\) is the efficiency of the implemented code, \(C(N)\) is the capacity of the Chien channel (eq. (4.22)) and \(N\) is the digital sum variation of the channel code. In our context, the most desirable code is maxentropic, and this will correspond with an efficiency of 100 percent.

As an example, let \(n = 4\) and \(K = 1\). From Table 5.3 we find in this case a digital sum variation and code rate \(N = 7\) and \(R = 0.83\), respectively, so that for this channel code an efficiency \(\eta = 0.83/0.886 = 95\%\) (see Table 4.1, page 78) is concluded. The sum variance of the maxentropic sequence with \(N = 7\) is 2.09 (see again Table 4.1). The sum variance of the implemented code is 1.5, which amounts to \(1.5/2.09 = 72\%\) of the sum variance of the maxentropic \((z)\) sequence with \(N = 7\). It is clear that the comparison of dc-balanced channel codes with maxentropic \((z)\) sequences should take into account both the sum variance and the code rate. We come to the following definition of encoder efficiency:

\[
E = \frac{(1 - C(n))\sigma_s^2(N)}{(1 - R)s^2(N)}. \tag{5.20}
\]

The efficiency \(E\), as defined in (5.20), compares the ‘redundancy-sum variance products’ of the implemented code and the maxentropic sequence with the same digital sum variation as the implemented code. Note that for \(N > 9\) the ‘redundancy-sum variance product’ of maxentropic \((z)\) sequences is approximately constant (see eq. (4.33)) and equals 0.2326. The efficiency \(E\) of various codes versus codeword length is plotted in Figure 5.3. Examination of the family of result curves highlights several important characteristics.

We observe that the low-disparity codes fall close to the performance of maxentropic sum constrained sequences provided the codeword length is small. As the block length \(n\) increases, the efficiency of the implemented codes diminishes. For example, when \(n = 2\) the
efficiency of the zero-disparity code format is 100%, it decreases to 60 percent when \( n = 20 \). Qualitatively, the same behaviour is found for low-disparity coding schemes.

The polarity bit encoding principle has the virtue of simple implementation, but, as we may notice from Figs. 5.2 and 5.3, it has a poor performance in terms of sum variance/redundancy product, and its performance is definitely far from optimum in the depicted range. The figures reveal that, for a given rate, the sum variance of a sequence encoded according to the polarity bit principle is typically 2.5 times that of maxentropic \((z)\) sequences. Eqs (4.33), (5.18) and (5.20) show that for large codewords the efficiency \( E \) asymptotically diminishes to 35 percent. In the range \( 0.8 < R < 0.9 \), notice for a desired width of the spectral notch the price of this simple code: a loss of approximately 15 percent in code rate. Figures 5.2 and 5.3, show the superiority of zero-disparity codes with respect to 'polarity bit' encoding in the most practical \((1 - R)\) interval. A calculation shows that for an unpractically large codeword length \( n > 160 \) the polarity bit encoding principle outperforms the zero-disparity encoding.
5.4 Practical Codes

In this section we provide a survey of dc-balanced codes that have found application in recorders and other products. The properties of the implemented codes shall be compared with the full set codes studied above.

5.4.1 5b6b Code

As an illustration, the spectrum of a code, called 5b6b code, can be used to evaluate the accuracy of the preceding analysis when the number of codewords are chosen to accommodate binary source words. The 5b6b code is basically an \( n = 6, K = 1 \), bi-mode code with six of the maximally available \( 20 + 2 \times 15 = 50 \) codewords deleted. In principle, any arbitrary six words can be discarded; we opted to delete the codewords containing a long runlength, namely '+++++-', '-++++-', '--++++' and their inverses.

![Figure 5.4](image)

**FIGURE 5.4** - Comparison of the spectra of 5b6b code and the bi-mode \( n = 6, K = 1 \) code.

A calculation, following the analysis provided in yields the power density function of the 5b6b code depicted in Figure 5.4. The power density function of the 'full-set' \( n = 6, K = 1 \) channel code, which can be obtained is plotted for comparison purposes. We observe a nice
agreement (a few dB difference) between the spectra of the implemented code and its full-set counterpart.

5.4.2 8b10b Code

In certain applications of channel codes, specifically digital recording on magnetic tape or transmission over fibre-optic network, it has been found that a rate $R = 8/10$, dc-balanced channel code has attractive features both in terms of system penalty and hardware realization. Most of the implementations in use are block codes which translate one byte into ten channel symbols. Clearly, a zero-disparity block code is impossible since the number of available 10-bit zero-disparity codewords, 252, is smaller than required. A two-state encoder offers the freedom of at maximum $252 + 210 = 462$ codewords. Since only 256 codewords are required this evidently offers a large variety of choices. The codebook can be tailored to particular needs such as minimum dc-content and/or ease of implementation. It is therefore not too surprising that in the literature, the patent literature in particular, numerous variants have been described.

<table>
<thead>
<tr>
<th>Reference</th>
<th>$N$</th>
<th>$T_{max}$</th>
<th>Note</th>
</tr>
</thead>
<tbody>
<tr>
<td>Morizzo [5]</td>
<td>10</td>
<td>10</td>
<td>**</td>
</tr>
<tr>
<td>Shirota [6]</td>
<td>7</td>
<td>6</td>
<td></td>
</tr>
<tr>
<td>Widmer [7],[8]</td>
<td>7</td>
<td>5</td>
<td></td>
</tr>
<tr>
<td>Parker [9]</td>
<td>6</td>
<td>5</td>
<td>***</td>
</tr>
<tr>
<td>Fukuda1 [10]</td>
<td>7</td>
<td>5</td>
<td></td>
</tr>
<tr>
<td>Fukuda2 [10]</td>
<td>7</td>
<td>4</td>
<td></td>
</tr>
</tbody>
</table>

Table 5.4 lists the main parameters of selected dc-constrained 8b10b and 16B20B codes that were documented in the literature. The Parker code is a 16b20b fixed-disparity code, and has the same main parameters, but due to the doubled codeword length it requires considerably more hardware for encoding and decoding than the other codes presented. The code designed by Widmer and Franaszek [8] is of interest due to its special structure. Each incoming byte is partitioned into two sub-blocks of five and three bits, respectively. Five binary input lines are encoded
into six binary output lines following the directions of the 5b6b look-up table and the disparity control. Similarly, the three remaining input bits are encoded into the remaining four output bits under the rules of a 3b4b look-up table and the disparity control. The 5b6b and 3b4b encoders operate to a large extent independently of each other; they operate on the basic principles of the bi-mode encoder. The number of available codewords is \((20 + 15) \times (6 + 4) = 350\). Modifications in the translation tables have been made to reduce the maximum runlength and the digital sum variation, to define special characters outside the data alphabet and to minimize implementation cost at high data rates. A trellis diagram of the encoder is shown in Figure 5.5. The figure reveals that the digital sum variation of the code is seven; there is no runlength longer than five. The authors reported that a 200 Mbaud fibre-optic link using this code has been built with ECL 10K-series circuits. It requires 107 gates for encoding, 79 gates for decoding, and 50 gates for a complete code violation check.

\[
\begin{array}{cccccccccc}
0 & 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 & 9 & 10 \\
\hline
3 & 0 & -1 & -2 & -3 & 0 & -1 & -2 & -3 & 0 & -1 \\
2 & 3 & 0 & -1 & -2 & -3 & 0 & -1 & -2 & -3 & 0 \\
1 & 2 & 3 & 0 & -1 & -2 & -3 & 0 & -1 & -2 & -3 \\
0 & 1 & 2 & 3 & 0 & -1 & -2 & -3 & 0 & -1 & -2 \\
\end{array}
\]

**FIGURE 5.5** - Unbalance trellis diagram of Widmer/Pranaszek code. The 8b10b is constituted by a 5b6b and a 3b4b encoder which operate to a large extent independently of each other; they operate on the basic principles of the bi-mode encoder. Note that the unbalance, or running digital sum, only takes two values at the symbol positions 0 and 6, respectively.

The Fukuda2 code [10] is employed as the recording code in the DAT digital audio tape recorder. The 8b10b code used in the R-DAT is designed to function well in the presence of crosstalk from neighbouring tracks, to allow the use of the rotary transformer and to have a small ratio of maximum and minimum runlengths to ease overwrite erasure. Essentially, the code operates according to the low-disparity principle; the encoder has two states. The codebook contains 153 zero-disparity codewords and 103 codewords of disparity \(\mp 2\). The hardware of the encoder and decoder has been reduced by computer optimizing the
relationship between the 8-bit source words and the 10-bit codewords. This has been done so that the conversion can be performed in a small programmed logic array. The codewords are stored using NRZI notation. The details of the look-up tables for encoding and decoding can be found in [10]. Only codewords of disparity 0 and +2 are produced by the logic, since the codewords of disparity -2 can be obtained by reversing the first symbol of the codeword. The maximum runlength is four and the sum variance of this code is 1.71. A variant of this 8b10b code was also reported by Fukuda et al., which possesses the virtue of reduced sum variance, 1.325, at the expense of a slightly increased maximum runlength.

5.5 References

Index

A

AlGaAs laser 7
alphabet of symbols 20
audio bits 3

B

Bernoulli 42
binary entropy function 22
bit clock 14
block code 50
Bowers 86

C

capacity 21, 41, 76
Cattermole 86
change-of-state encoder 38
Chien 75, 102
Code
Bi-Phase 39, 100
CIRC 4
EFM 3, 38, 39, 50
FM 39
MFM 39, 52
Miller 39
Reed-Solomon 4
3PM 39
5b6b 104
8b10b 105
(0,3) Code 63
(1,7) Code 67
(2,7) Code 65
code rate 21, 41
codebook 11
codeword assignment 51
Compact Disc 3
connection matrix 31, 45, 75
constrained channel 21
coupling components 69
cut-off frequency 73
cyclo-stationary 94
C&D 4

d

DAT recorder 106
decomposition 27
density ratio 43
detection window 37, 44
deterministic constraint 21
disparity 85
dk constraint 38
DSV 75

E

EFM coding table 17
entropy 22, 29, 76
enumerative encoding 60
equilibrium distribution vector 26
ergodic chain 25
error burst 4
error propagation 50, 60
eye pattern 7
Index

F

Fibonacci numbers 40
fingermark 15, 69
finite-state machine 45
fixed-length code 50
Fourier analysis 10
frame 4
Franaszek 53
Fukuda 105
future-dependent coding 67

I

Immink 105
information content 20
information density 6
information source 20
input-restricted 21, 45
interleaving 4, 36

J

jitter 37
Justesen 93

L

Lagrange multipliers 31
laser 3
lexicographical order 60
look-ahead coding 67
low-disparity 88
low-frequency content 14

M

magnetization 38
manufacturing tolerances 7
Markov chain 24
Markov condition 24
Markov process 91
masking of errors 4
master disc 3, 7
maxentropic 46, 80
maximum runlength 14, 37
Mealy machine 28
merging bit 4, 52
merging rule 52
minimum runlength 11, 37
Moivre 42
Moore machine 28
Morizono 105
multiplexer 6

N

nat(ural unit) 22
noiseless channel 21
NRZ 38
NRZI 38, 107
numerical aperture 7
Nyquist's sampling theorem 3

O

overwrite erasure 106

P

packing density 43
parity symbol 4
Parker 105
Perron-Frobenius theorem 31
phase jitter 7
polarity bit 86, 98
polynomial 77
prefix condition 65
principal state 53
pulse-code modulation 5

R

random-walk 75
RDS 18, 89
regular chain 25
RLL 37
runlength 37
runlength constraint 37
runlength distribution 47

S
self-punctuating 65
servo 14
Shannon 22, 29, 45
Shirota 105
spectral notch 71
spot diameter 7
state alphabet 24
steady state probability 26
stochastic matrix 24
subcode 4
sum variance 78
synchronization 5, 63

T
Toeplitz matrix 75
trellis diagram 25, 91

U
uncertainty 22, 29

V
variable-length synchronous code 64

W
waveform 20, 38
wavelength 7
weighting system 62
Widmer 105
Winchester drive 52

Z
z-constrained sequences 75
zero-disparity 85, 87

( (dklr) sequence 62
(dk) sequence 38
(z) sequence 75