Runlength-Limited Sequences

KEES A. SCHOUHAMER IMMINK, FELLOW, IEEE

Coding techniques are used in communication systems to increase the efficiency of the channel. Not only is coding equipment being used in point-to-point communication channels, but coding methods are also used in digital recording devices such as sophisticated computer disk files and numerous domestic electronics such as stationary- and rotary-head digital audio tape recorders, the Compact Disc, and floppy disk drives. Since the early 1970s, coding methods based on runlength-limited sequences have played a key role for increasing the storage capacity of magnetic and optical disks or tapes. A detailed description is furnished of the limiting properties of runlength-limited sequences, and a comprehensive review is given of the practical aspects involved in the translation of arbitrary data into runlength-limited sequences.

I. Introduction

Given the more or less constant demand for increased storage capacity and reduced access time, it is not surprising that interest in coding techniques for optical and magnetic recorders has continued unabated ever since the day when the first mechanical computer memories were introduced. Naturally, technological advances, notably of a mechanical nature, have greatly increased the storage capacity, but state-of-the-art storage densities are also a function of improvements in channel coding.

Coding techniques are used in communication systems to improve the reliability and efficiency of the communication channel. The *reliability* is commonly expressed in terms of the probability of receiving the wrong information, that is, information that differs from what was originally transmitted. Error control is concerned with techniques of delivering information from a source (the sender) to a destination (the receiver) with a minimum of errors. As such it has its roots in information theory and goes back to the pioneering work of Shannon. Information theory partitions the coding problem into two main categories: *source* and *channel* coding. Source coding is, roughly speaking, a technique to reduce the source symbol rate by removing the redundancy in the source signal. A convenient definition of channel coding is: The technique of realizing high transmission reliability despite shortcomings of the channel, while making efficient use of the channel capacity. In essence, the tenets of information theory show that a stationary channel can be made arbitrarily reliable given that a fixed fraction of the channel is used for redundancy. This raises the immediate question: How can the promised theoretical performance be attained in practice? There is no answer to that question at this moment; there is, however, a *de facto* consensus as to which code structure is "good." In recorder systems, channel coding is commonly accomplished in two successive steps: a) error-correction code and b) recording code. The various coding steps are visualized in Fig. 1 which shows a block diagram of a possible recording system of this kind.

Error-correction control is realized by systematically adding extra symbols to the conveyed message. These extra symbols make it possible for the receiver to detect and/or correct some of the errors that may occur in the received message. The main problem is to achieve the required protection against the inevitable transmission errors without paying too high a price in adding extra symbols. There are many different families of error-correcting codes. Of major importance for recording applications is the family of *Reed–Solomon codes*. The reason for their preeminence in recording systems is that they can combat combinations of random as well as burst errors.

The arrangement termed *recording code* (sometimes referred to as line or modulation code) on which this paper will concentrate, accepts the bit stream (extra bits added by the error-correction system included) as its input and converts the stream to a waveform suitable for the specific recorder requirements. The object of the recording code is to bring structure into a data stream that is generally not present in the information supplied by the user. All the aforementioned coding stages are present in a modern digital video recorder [1].

Schematically the elements of the coding steps in a digital recorder as shown in Fig. 1 are similar to those of a "point-to-point" communication link. For example, in a radio or satellite link the encoded information is used to modulate a carrier. In this context, the method of putting a second code on top of the first one, as seen in the structure of Fig. 1, is usually called a *concatenated scheme*.

The physical constraints dictated by the channel lead to the simple fact that some sequences have to be discarded, and therefore not the entire population of sequences can be used. For instance, common magnetic recorders will not respond to low-frequency signals, so that in order to minimize distortion of the retrieved data, one usually eliminates low-frequency components in the recorded data by

Manuscript received November 3, 1989; revised July 19, 1990.
The author is with Philips Research Laboratories, 5600 JA Eindhoven, The Netherlands.
IEEE Log Number 9039185.

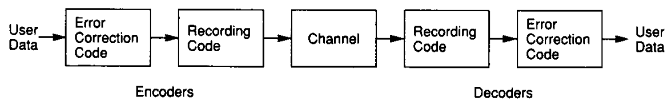


Fig. 1. Block diagram of a recording system. A data source generates user data in the form of (binary) symbols. The source data is translated in two successive steps: a) error-correction code and b) recording code. The output generated by the recording code is stored on the storage medium in the form of binary physical quantities, for example, pits and lands, or positive and negative magnetizations. During read-out, the source data are obtained via the decoder for the recording code and the error-correction code.

a coding step. The likelihood of errors is reduced by avoiding those sequences which are *a priori* known to be most vulnerable to channel impairment, and thus prone to error. The actual process of selecting and devising a recording code is in fact always a compromise between conflicting requirements. The choice of a particular code depends on numerous factors such as available signal-to-noise ratio, clocking accuracy, non-linearities, and intersymbol interference (time varying!). A recording code must show a certain robustness against dynamic changes of the channel's characteristics. This statement applies specifically for the DAT and Compact Disc, which are meant for domestic use, and where, thus readjustment of the mechanical parameters is out of the question. Other constraints, such as equipment limitations, ease of encoding and decoding, and the desire to preserve a particular mapping between the source and the code symbols all govern to a greater extent the encoding chosen. Nontechnical considerations such as the patent situation or familiarity (prejudice?) may also influence the choice and are to some extent certainly valid. One further, and certainly not the least, factor that affects the choice of coding scheme is its cost: any coding scheme is merely a part of a larger system and its cost must be in proportion to its importance within that system.

An encoder has the task of translating arbitrary user information plus error-correction symbols into a sequence which conforms to the given channel constraints. On the average m binary user symbols are translated into n binary channel symbols. Obviously, since a restricted repertoire is used, $n > m$. A measure of the efficiency implied by a particular code is given by the quotient $R = m/n$, where R is called the *rate* of the code. The encoder translates user information into the channel representation using a kind of dictionary or look-up table. The problems involved are not trivial. The straightforward implementation would require look-up tables of unacceptably large size when the codewords are comparatively long. This difficulty is circumvented by developing algorithmic procedures that allow the encoder to compute a codeword from the user information. In turn, the decoder processes the retrieved waveforms and also performs the inverse mapping, using a similar algorithmic routine, to the digital output sequence. Theoretical and practical contributions are still being made to the subject, which is indicative of its hidden complexities.

Recording codes that are based on *runlength-limited* (RLL) sequences have found almost universal application in optical and magnetic disk recording practice. The length of time, usually expressed in the number of binary symbols, between consecutive transitions is known as the *runlength*. Runlength-limited 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 has a bearing on the intersymbol interference when the sequence is conveyed over a bandwidth-limited channel. If the transitions are too close, the intersymbol interference between adjacent symbols may become excessive. In RLL sequences, the occurrence of such destructive patterns is avoided. 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 information density, and the jitter and noise characteristics. Archetypal RLL codes are the rate 1/2, $(d = 2, k = 7)$ code which is applied in the IBM3380 rigid disk drive [2], [3], and the Eight-to-Fourteen Modulation (EFM), a rate 8/17, $(d = 2, k = 10)$ code, which is the basis of the Compact Disc [4]. Runlength-limited codes, in their general form, were pioneered in the 1960s by Freiman and Wyner [5], Kautz [6], Gabor [7], Tang and Bahl [8], and notably Franaszek [9]. It is undoubtedly the case that RLL codes have generated a high and sustained level of interest amongst workers ever since the introduction of the basic ideas, and there is now a considerable amount of literature available on the design of encoding and decoding devices of RLL sequences.

The outline of this paper is as follows. We commence with a formal definition of RLL sequences, followed by an analysis of the runlength distribution and spectral properties of ideal RLL sequences. In the remaining sections, examples of code implementation are presented to illustrate the variety of possible design tools.

II. Definition of Runlength-limited Sequences

The theory on RLL 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 an RLL channel sequence in the following way. Let the channel signals be represented by a sequence $\{y_i\}, y_i \in \{-1, 1\}$. 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 $1 \to -1$ or $-1 \to 1$ of the corresponding RLL sequence. The (dk) sequence

$$0\,1\,0\,0\,0\,1\,0\,0\,1\,0\,0\,0\,1\,1\,0\,1\cdots$$

would be converted to the RLL channel sequence

$$1\ -1\ -1\ -1\ -1\ 1\ 1\ 1\ -1\ -1\ -1\ -1\ 1\ -1\ -1\ 1\ 1\cdots$$

The mapping of the waveform by the precoding step is known as the non-return-to-zero-inverse (NRZI) data encoding method. Waveforms that are transmitted without such an intermediate coding step are referred to as non-return-to-zero (NRZ). The nebulous terms NRZ and NRZI stem from telegraphy, and have 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.

III. Number of (dk) Sequences

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. Define

$$N_d(n) = 0, \quad n < 0,$$
$$N_d(0) = 1. \tag{1}$$

The number of (d) sequences of length $n > 0$ is found with the recursive relations [8]

$$N_d(n) = n + 1, \quad 1 \le n \le d + 1,$$
$$N_d(n) = N_d(n-1) + N_d(n-d-1), \quad n > d + 1. \tag{2}$$

Table 1 lists the number of distinct (d) sequences as a function of the sequence length n with the minimum runlength d as a parameter. Recursion relation (2) has an elementary interpretation. Let $(x_1, x_2, \cdots, x_{n-1})$ be a (d) sequence of length $n - 1 > d$, then $(x_1, x_2, \cdots, x_{n-1}, 0)$ and $(x_1, x_2, \cdots, x_{n-d-1}, 0^d, 1)$ 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 $N_d(n-1) + N_d(n-d-1)$.

When $d = 0$, we simply find that $N_0(n) = 2N_0(n-1)$, 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 $N_1(n)$ are 1, 2, 3, 5, 8, 13, $\cdots$, 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 [10]. The ratio of two successive Fibonacci numbers $N_1(n+1)/N_1(n)$ approaches for large n the golden ratio $g = (1 + \sqrt{5})/2$. 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

$$N(n) = 0, \quad n < 0,$$
$$N(0) = 1. \tag{3}$$

The number of (dk) sequences of length n is given by [8]

$$N(n) = n + 1, \quad 1 \le n \le d + 1,$$
$$N(n) = N(n-1) + N(n-d-1), \quad d + 1 \le n \le k,$$
$$N(n) = d + k + 1 - n + \sum_{i=d}^{k} N(n-i-1),$$
$$k < n \le d + k,$$
$$N(n) = \sum_{i=d}^{k} N(n-i-1), \quad n > d + k. \tag{4}$$

The k-limited case, $d = 0$, can be derived as a special case of the general dk case. It should be appreciated that the d, k, and dk sequences defined above cannot in general be cascaded without the risk of violating the specified constraints at the boundaries. In a later section, we will describe methods of inserting buffering (merging) sequences that allow the concatenation of codewords without violating the dk constraints

IV. 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. What is the maximum value of $R = m/n$ that can be attained for some specified values of the minimum and maximum runlength d and k? The answer was provided by Shannon [11]. The maximum value of R that can be achieved is termed the *capacity*. The capacity, denoted by $C(d, k)$, is governed by the specified constraints and is given by

$$C(d, k) = \lim_{n\to\infty} \frac{1}{n} \log_2 N(n). \tag{5}$$

Table 1 Number of Distinct (d) Sequences as a Function of the Sequence Length n and the Minimum Runlength d as a parameter

$n =$	4	5	6	7	8	9	10	11	12	13	14
$d = 1$	8	13	21	34	55	89	144	233	377	610	987
2	6	9	13	19	28	41	60	88	129	189	277
3	5	7	10	14	19	26	36	50	69	95	131
4	5	6	8	11	15	20	26	34	45	60	80
5	5	6	7	9	12	16	21	27	34	43	55

For notational convenience we restrict ourselves for the time being to (d) sequences. Equation (5) 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 (2). According to (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

$$z^{d+1} - z^d - 1 = 0. \tag{6}$$

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 λ_i, $i = 1, \cdots, d+1$, are the roots of (6), and a_i are constants, independent of n, to be chosen to meet the first $(d+1)$ values of $N_d(n)$. If $\lambda = \max_i \{\lambda_i\}$ is the largest real root of (6), then for large n,

$$N_d(n) \propto \lambda^n.$$

Applying definition (5), the asymptotic information rate of d-constrained sequences, denoted by $C(d, \infty)$, is

$$C(d, \infty) = \lim_{n \to \infty} \frac{1}{n} \log_2 N_d(n) = \log_2 \lambda. \tag{7}$$

This is the sought-after result we need: the quantity $C(d, \infty)$ provides the maximum rate possible of any implemented code given the channel constraints.

Example (1): Let $d = 1$, then we obtain the characteristic equation

$$z^2 - z - 1 = 0,$$

with solutions

$$\lambda_1 = \tfrac{1}{2}(1 + \sqrt{5}) \quad \text{and} \quad \lambda_2 = \tfrac{1}{2}(1 - \sqrt{5}).$$

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:

$$N_1(n) = \frac{1}{\sqrt{5}}\left(\frac{1+\sqrt{5}}{2}\right)^{n+2} - \frac{1}{\sqrt{5}}\left(\frac{1-\sqrt{5}}{2}\right)^{n+2}$$

$$= \frac{1}{\sqrt{5}}\{g^{n+2} - (-g)^{-n-2}\}, \quad n \geq 0. \tag{8}$$

Thus λ equals the golden ratio $\lambda = g = (1 + \sqrt{5})/2$, and the capacity is

$$C(1, \infty) = \log_2 \frac{1+\sqrt{5}}{2} \simeq 0.694.$$

□

Table 2 collects further values of the capacity for different values of d.

The quantity DR, called *density ratio*, is sometimes referred to as *packing density*. It expresses the minimum physical distance between consecutive transitions of an RLL sequence given the information rate is fixed. It is defined as

$$DR = (1 + d)R, \tag{9}$$

Table 2 Capacity and Density Ratio DR versus Minimum Runlength d

d	$C(d, \infty)$	$(d+1)C(d, \infty)$
1	0.694	1.388
2	0.551	1.654
3	0.465	1.860
4	0.406	2.028

where R is the rate of the RLL code. It can be seen from Table 2 that an increase of the density ratio is obtained at the expense of decreased capacity. It may be shown that the density ratio DR is made arbitrarily large by choosing the minimum runlength d sufficiently large. This follows from

$$DR = (d + 1) \log_2 \lambda.$$

The root λ of (6) satisfies

$$\left(\frac{(1-\epsilon)d}{\log_2 d}\right)^{1/d} \leq \lambda \leq \left(\frac{(1+\epsilon)d}{\log_2 d}\right)^{1/d}$$

for large d. Thus DR grows like a constant times $\log d$, see also Table 2. Codes with a larger value of d, and thus a lower rate, are penalized by an increasingly difficult trade-off between the detection window (the size of a channel bit) and the density ratio in applications with very high information density and data rates.

In similar vein to the case of d-constrained sequences, it is possible to derive the capacity $C(d, k)$, k finite. The capacity $C(d, k)$ is the base-2 logarithm of the largest real root of the characteristic equation of (dk) sequences [8]

$$z^{k+2} - z^{k+1} - z^{k-d+1} + 1 = 0. \tag{10}$$

Table 3 lists the capacity $C(d, k)$ versus the runlength parameters d and k.

A. State-Transition Matrix Description

There is an alternative technique to derive the channel capacity, which is based on the representation of the dk constraints by a finite-state sequential machine. Fig. 2(a) illustrates a possible state-transition diagram. There are $(k+1)$ states which are denoted by $\{\sigma_1, \cdots, \sigma_{k+1}\}$. Transmission of a "zero" takes the sequence from state σ_i to state σ_{i+1}. A "one" may be transmitted only when the machine occupies states $\sigma_{d+1}, \cdots, \sigma_{k+1}$. Any path through the state-transition diagram defines an allowed (dk) sequence. The adjacency or connection matrix, which gives the number of ways (paths) of going (in one step) from state σ_i to state σ_j, is given by the $(k+1) \times (k+1)$ array D with entries d_{ij}, where

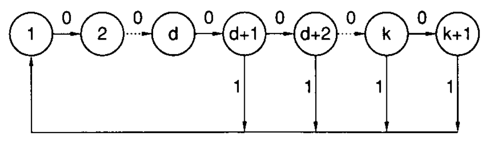
$$\begin{aligned} d_{i1} &= 1, \quad i \geq d+1, \\ d_{ij} &= 1, \quad j = i+1, \\ d_{ij} &= 0, \quad \text{otherwise}. \end{aligned} \tag{11}$$

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

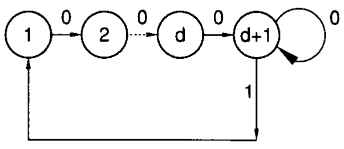
$$D = \begin{bmatrix} 0 & 1 & 0 & 0 \\ 1 & 0 & 1 & 0 \\ 1 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 \end{bmatrix}. \tag{12}$$

Table 3 Capacity $C(d, k)$ versus Runlength Parameters d and k

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



(a)



(b)

Fig. 2. (a) State-transition diagram for a (dk) sequence. Transmission of a "zero" takes the sequence from the state σ_i to state σ_{i+1}, $i \leq k$. A "one" may be transmitted only when the machine occupies states $\sigma_{d+1}, \cdots, \sigma_{k+1}$, while a "one" must be produced if the machine is in state σ_{k+1}. The machine returns to state σ_1 after transition of a "one." (b) State-transition diagram for a (d) sequence. Only in state σ_{d+1} both a "zero" and a "one" may be emitted, in all other states a "zero" must be emitted.

A d-constrained channel can be modeled with $d + 1$ states (see Fig. 2(b)). The connection matrix is given by the $(d + 1) \times (d + 1)$ array D with entries d_{ij}, where

$$d_{ij} = 1, \quad j = i + 1,$$
$$d_{d+1,1} = d_{d+1,d+1} = 1,$$
$$d_{ij} = 0, \quad \text{otherwise.} \tag{13}$$

The above representation is an example of the input-restricted noiseless channel studied by Shannon [11]. 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. The number of distinct sequences of length n that emanate from state σ_i and terminate in state σ_j is given by the ijth entry of D^n. The matrix D^{10} of the $(d, k) = (1, 3)$ constrained channel is

$$D^{10} = \begin{bmatrix} 17 & 11 & 8 & 5 \\ 24 & 17 & 11 & 8 \\ 19 & 13 & 9 & 6 \\ 11 & 8 & 5 & 4 \end{bmatrix}. \tag{14}$$

It can be seen, for instance, that there are exactly 19 sequences of length 10 that emanate from state σ_3 and terminate in state σ_1.

An alternative way to express the capacity of a finite-state machine that emits dk-constrained sequences is

$$C(d, k) = \lim_{n\to\infty} \frac{1}{n} \log_2 \sum_{i,j} [D]^n_{ij}, \tag{15}$$

where $[D]^n_{ij}$ denotes the ijth entry of D^n. Shannon [11] showed that

$$C(d, k) = \log_2 \lambda, \tag{16}$$

where λ is the largest real root of the characteristic equation

$$\det [D - zI] = 0, \tag{17}$$

where I is the identity matrix.

V. Spectrum of Ideal RLL Sequences

The power density function, or in short, spectrum, of an RLL sequences offers a measure of the bandwidth occupation, and it provides a means of assessing the interference induced from signals from adjacent tracks. To compute the spectrum of RLL sequences generated by implemented codes can be a difficult task. It is, however, possible to find the spectrum if some simplifying assumptions regarding the runlength distribution are made. To this end, consider a source that emits a sequence of binary symbols whose runlengths $T_i \in \{d + 1, \cdots, k + 1\}$ are i.i.d. variables. Let the probability of occurrence of runlength T_i be denoted by $Pr(T_i)$, then the power spectral density function can be expressed by [12]:

$$H(\omega) = \frac{1}{\overline{T} \sin^2 \omega/2} \frac{1 - |G(\omega)|^2}{|1 + G(\omega)|^2}, \tag{18}$$

where

$$G(\omega) = \sum_{l=d+1}^{k+1} Pr(T_l) e^{j\omega l}, \quad j = \sqrt{-1}, \tag{19}$$

and

$$\overline{T} = \sum_{l=d+1}^{k+1} l Pr(T_l). \tag{20}$$

Using information theoretical arguments, it can be shown that an ideal RLL sequence, that is, a sequence with maximum information content, has a truncated geometric runlength distribution with parameter λ [13], [14], [15]. Then

$$Pr(T_i) = \lambda^{-T_i}, \quad T_i = d + 1, \cdots, k + 1, \tag{21}$$

where λ is the largest real root of (10). Substitution of the distribution (21) provides a straightforward method of determining the spectrum of ideal RLL sequences. Fig. 3 depicts the spectra $H(\omega)$ of some ideal (d) sequences for various values of the minimum runlength d. 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.

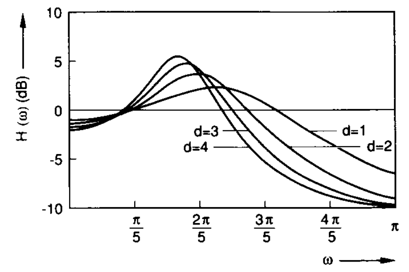


Fig. 3. Power density function versus frequency of ideal runlength-limited sequences.

We may observe the following characteristics: maxima occur at nonzero 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.

VI. Practical Coding Schemes

In the previous sections, properties, such as capacity and spectra, of ideal RLL sequences were 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 RLL sequences in a practical manner. It is most important that this be done as efficiently as possible within some practical considerations. The previous sections on ideal RLL sequences provide the cardinal limits. Codes that have found application outside the laboratory apply to RLL codes with parameters $d = 0, 1$, and 2, which is, of course, reflected in the survey.

A. Fixed-Length (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 encoder chops the source sequence into blocks of length m, and under the code rules such blocks are mapped onto words of n channel symbols. The obvious method for the construction of RLL codes is to employ codewords that can be freely cascaded without violating the sequence constraints; the codewords employed have a one-to-one correspondence with the source words. A logical step, then, is to extend this mechanism to codes, where the translations are not one-to-one, but are, for example, a function of the latest codeword transmitted. The source words operate with a number of alternative translations (or modes); each of them is interpreted by the decoder in the same way. During transmission, the choice of a specific translation is made in such a way that the specified constraints of the encoded sequence, after transmission of the new codeword, are not violated. Codes that operate with a unique representation of the source words are called state-independent codes, and codes whose translations are not one-to-one are called state-dependent codes. The restriction of state-independent encoding leads, in general, to codes that are more complex than state-dependent codes for a given bit-per-symbol value. To clarify this concept we have written down a single illustrative case of a (1, ∞) code.

The codeword assignment of Table 4 provides a simple block code that converts source blocks of length $m = 3$ onto codewords of length $n = 5$. The two columns furthest to the left tabulate the eight possible source words along with their decimal representation. The codewords, tabulated in the right hand column, can be freely cascaded without violating the $d = 1$ constraint, since 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 $d = 1$ constraint (see also Table 1).

Table 4 Simple Fixed-Length $d = 1$ Code

source		*output*
0	000	00000
1	001	00001
2	010	00010
3	011	00100
4	100	00101
5	101	01000
6	110	01001
7	111	01010

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

$$\eta = \frac{R}{C(d, k)} \simeq \frac{0.6}{0.69} \simeq 0.86. \tag{22}$$

This, actually, demonstrates that good efficiencies are feasible with simple constructions. That this example is not untypical will be demonstrated in the remainder of this section. 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. A systematic approach of choosing a specific assignment that simplifies the look-up function is not available, and only little progress has been made [16].

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 conditions is $N_d(n - d)$, which can be computed with (2) or by using Table 1.

A maximum runlength constraint can be incorporated in the code rules in a straightforward manner. For instance, in the $(1, \infty)$ block code listed in Table 4, 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 first symbol, the *merging rule*, can be implemented quite smoothly with some extra hardware. It is readily conceded that with this additional merging rule the $(1, \infty)$ code, presented in Table 4, turns into a (1, 6) code. The efficiency of the (1, 6) code is now, as can be verified with Table 3: $\eta = 0.6/C(1, 6) \simeq 0.6/0.669 \simeq 0.897$. The process of decoding is exactly the same as for the simple $(1, \infty)$ code, since the first bit, the merging bit, was skipped anyway. The (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).

1) Modified Frequency Modulation: Modified Frequency Modulation (MFM), a rate = 1/2, (1, 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 5. The symbol indicated

Table 5 Coding Rules MFM Code

source	*output*
0	x0
1	01

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 $k = 3$.

MFM has high efficiency, $\eta \simeq 0.5/0.5515$, or approximately 91 percent. A graphical representation of the finite-state machine underlying the MFM code, using the NRZI notation rules, is pictured in Fig. 4. The labelled edges ema-

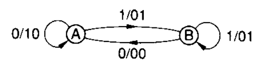


Fig. 4. Two-state transition diagram that describes the MFM code.

nating from a state define the encoding rule, and the state in which an edge terminates indicates the state, or coding rule to use next. State A represents the condition that the previous channel bit was a "zero," while 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.

B. 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 any 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 [9] developed a technique of successive elimination for determining the existence of a set of principal states through operations on the connection matrix. The subsequent procedure, taken from [17], 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 $\Sigma = \{\sigma_i\}$. Let further Σ^* be the set of states that have not been eliminated yet and $\sigma_i \in \Sigma^*$ a state to be tested by the algorithm. The number $\psi(\sigma_i, \Sigma^*)$ of (dk) sequences of length n permitted from σ_i and terminating in a state $\sigma_j \in \Sigma^*$ is given by

$$\psi(\sigma_i, \Sigma^*) = \sum_{j \in V} [D]^n_{ij}, \qquad V = \{ j: \sigma_j \in \Sigma^*\}, \tag{23}$$

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

$$\psi(\sigma_i, \Sigma^*) \geq 2^m,$$

thus Σ^* is the set of principal states. The foregoing analysis which is due to Franaszek, can be cast into a form that is of use later. We introduce an *approximate eigenvalue inequality* to guide the construction [18].

Let $v = (v_1, \cdots, v_{k+1})^T$, $v_i \in \{0, 1\}$, be a vector with binary elements. A fixed-length code of the specified runlength constraints and parameters can be ascertained if there is a binary vector v that satisfies

$$D^n v \geq 2^m v. \tag{24}$$

In our context, the vector v is usually called the *approximate eigenvector*. For a given code rate $R = m/n \leq C(d, k)$, the existence of such an eigenvector is guaranteed by the Perron-Frobenius theory of nonnegative matrices [19]. It is not difficult to see that the set $\{\sigma_{j_1}, \cdots, \sigma_{j_L}\}$ for which $v_{j_1} = \cdots = v_{j_L} = 1$ 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 2: We examine here the construction of a (1, 3) code. Table 3 indicates that a code rate 1/2 represents 90% of the channel capacity. Therefore, let $m = 1$ and $n = 2$. The matrix D^2 is

$$D^2 = \begin{bmatrix} 1 & 0 & 1 & 0 \\ 1 & 1 & 0 & 1 \\ 1 & 1 & 0 & 0 \\ 0 & 1 & 0 & 0 \end{bmatrix}. \tag{25}$$

Use of the successive elimination algorithm indicates that state σ_4 has to be deleted (the row sum for row four is only one). We eliminate row four and column four. The 3×3 submatrix so obtained has row sums which are exactly two. Thus the principal states are σ_1, σ_2, and σ_3. The codewords available for encoding associated with the principal states are

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

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

$$W(\sigma_3) = \begin{cases} 01 \\ 10. \end{cases} \tag{26}$$

A state-independently decodable code can be constructed with the assignments

$W(\sigma_1)$	$W(\sigma_2)$	$W(\sigma_3)$
01	01	01
00	10	10

After some rearrangement we obtain the following simplified codebook:

source	*output*
0	x0
1	01

A comparison with Table 5 reveals that this is the MFM code.

□

Example 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 2)

$$C(2, \infty) \simeq 0.551.$$

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. One can prove that the shortest fixed-length code with rate = 1/2 has a codeword length of 14. To this end, 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

$$\sum_{j=1}^{3} [D]_{1j}^{14} = 129$$

$$\sum_{j=1}^{3} [D]_{2j}^{14} = 189$$

$$\sum_{j=1}^{3} [D]_{3j}^{14} = 277.$$

Note that indeed in any state more than $2^m = 2^7 = 128$ sequences start and end that comply with the channel restrictions. 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. In fact, only one terminal state, namely σ_3, suffices for encoding. Apparently, this particular code can be state-independently encoded and decoded (all sequences start and end in state σ_3). Actually, this outcome is not surprising, for, when we take a look at Table 1, 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), (2, ∞) block code.

□

The shortest fixed-length codes for a selection of (d, k) combinations were computed by Franaszek [20] with the successive elimination procedure; the results are collected in Table 6. We draw attention to the fact that the minimum

Table 6 Shortest Fixed-Length Block Codes of Given Bit-Per-Symbol Values for a Selection of (d, k) Constraints, After Franaszek 1970, [20]

d	k	m	n	$\eta = R/C(d, k)$
0	1	3	5	0.864
0	2	4	5	0.910
0	3	9	10	0.951
1	3	1	2	0.907
1	7	11	33	0.981
2	5	4	10	0.860
2	7	17	34	0.962
2	10	8	16	0.923
3	7	46	115	0.986
3	11	8	20	0.886
4	9	9	27	0.921
4	14	12	33	0.916
5	12	9	30	0.890
5	17	15	45	0.937

codeword lengths of the fixed-length $R = 2/3$, (1, 7) code and the $R = 1/2$, (2, 7) code are 33 and 34, respectively. For these specific cases, a design with variable-length codewords (see later) provides shorter word lengths and reduced complexity.

C. Fixed-Length Codes Based on (dklr) Sequences

The previous construction technique has the obvious drawback that we require a large number of look-up tables for encoding. As a matter of fact, the number of look-up tables equals, in general, 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.

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

a) l constraint: the number of consecutive leading "zeros" of the sequence is at most l,
b) r constraint: the number of consecutive trailing "zeros" of the sequence is at most r.

The additional constraints imposed on the number of "zeros" at the beginning and end of the sequence allow the design of more efficient block codes than provided by the technique of Tang and Bahl [8].

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 β 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 describe 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 $r + d + l \le k$ 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 β merging bits are all set to "zero."

Construction 2: Choose d, k, and n' such that $k \ge 2d$. 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 merging bits are governed by the following rules. Let an n'-sequence end with a run of s "zeros" ($s \le r$) while the next sequence starts with $t (t \le l)$ leading "zeros." Table 7 shows the merging rule for the $\beta = d$ merging bits.

Table 7 Merging Rules of $(dklr)$ Sequences

s, t	Merging bits
$s + t + d \le k$	0^d
$s + t + d > k$	
if $s \le d$	$0^{d-s}10^{s-1}$
if $s > d$	10^{d-1}

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 $d = 1, \cdots, 4$ and $k = 2d, \cdots, 20$ we have selected $n = n' + d$ in such a way that the information rate R was maximized.

Tables 8 and 9 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% 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 favor of Construction 2. With Franaszek's algorithm it can be verified that the codes presented in Table 9 are of minimum length.

When the code rate R approaches the channel capacity $C(d, k)$, fixed-length code constructions have the drawback, as seen in the preceding examples, that the implementations can be very complex, involving long codewords.

Table 8 Fixed-Length Block Codes based on Construction 1

d	k	n	R	$\eta = R/C(d, k)$
1	7	12	8/13	0.91
2	17	14	8/16	0.91
3	14	17	8/20	0.87
4	18	19	8/23	0.87

Table 9 Fixed-Length Block Codes based on Construction 2

d	k	n'	R	$\eta = R/C(d, k)$
1	5	12	8/13	0.95
2	10	14	8/16	0.92
3	10	17	8/20	0.90
4	12	19	8/23	0.90

D. 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 could easily become an engineering impossibility. We can, however, create codewords by an algebraic procedure, called *enumerative encoding*, which means that there is no need to store 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 and $(dklr)$ 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 $0, 1, \cdots, |T(d, n)| - 1$, 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 $\mathbf{x} = (x_{n-1}, \cdots, x_0) \in T$ and $\mathbf{y} = (y_{n-1}, \cdots, y_0) \in T$, then $\mathbf{y}$ is called less than $\mathbf{x}$, in short, $\mathbf{y} < \mathbf{x}$, 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 $\mathbf{x}$ in the lexicographical ordering of T is defined to be the rank of $\mathbf{x}$ denoted by $r(\mathbf{x})$, i.e., $r(\mathbf{x})$ is the number of all $\mathbf{y}$ in T with $\mathbf{y} < \mathbf{x}$.

Theorem 1: The rank $r(\mathbf{x})$ of the (d) sequences $\mathbf{x} \in T$ can be calculated according to

$$r(\mathbf{x}) = \sum_{j=0}^{n-1} N_d(j) x_j. \tag{27}$$

Proof: See [21].

Example 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:

$r(\mathbf{x})$	$\mathbf{x}$
0	0000
1	0001
2	0010
3	0100
4	0101
5	1000
6	1001
7	1010

□

The inverse function, conversion from a given integer I to a (dk) sequence with rank I can be carried out as follows.

Inverse Algorithm: Let the set $T(d, n)$ and an integer I, $I = 0, 1, \cdots, |T(d, n)| - 1$, be given. The following algorithm finds $\mathbf{x}$ such that $r(\mathbf{x}) = I$.

```
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 1 allows the realization 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 $N_d(n-1), N_d(n-2), \cdots$. Storage capacity is required for approximately n nonzero weighting coefficients, a full adder, and an accumulator to store the intermediate and final results. The hardware requirements have to be compared with a look-up table of 2^m entries if a nonalgebraic method for coding is used. The use of a look-up table obviously sets a practical limit to the codeword length which can be decoded. The limit depends on technology and required bit rate, but 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 to 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 $\mathbf{x}$ 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 practical to do encoding and decoding by means of buffering and a large number of hard-wired adders.

The preceding algorithms for enumerating codewords can be generalized to the encoding and decoding of (dk) sequences and $(dklr)$ sequences.

E. Examples of Code Implementation

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

Rate 4/5, (0, 2) Code

The Group-Coded Recording (GCR) code, also known as "4/5 code," is used in a large variety of magnetic tape products. In the GCR code, four user bits are uniquely represented by five channel bits. The constraints placed upon the code are $d = 0$ and $k = 2$. According to Table 3, the capacity of a sequence with no runs of more than two "zeros" is $C(0, 2) \simeq 0.879$. Thus the efficiency of the GCR code is $\eta = 91\%$. The GCR code is a straightforward application of Construction 1. From the 32 possible combinations of five bits, 15 are eliminated because of the (0, 2) constraints, leaving 17, from which one can be discarded to produce the 16 unique patterns required. This one can then be used as a special pattern, for checking or error detection, as it obeys the specified constraints. A simple look-up table is used for encoding and decoding.

3 PM Code

The basic parameters of the 3PM (Three Position Modulation) code, which was invented by Jacoby [22], are $d = 2$, $k = 11$ and $R = 1/2$. The encoding mechanism of the 3PM code is similar to fixed-length block encoding and decoding with one extra rule. The encoding is explained by looking at Table 10. As we may notice, the right-hand boundary

Table 10 Basic Coding Table 3PM Code

Data	*Code*
000	000010
001	000100
010	010000
011	010010
100	001000
101	100000
110	100010
111	100100

position is occupied by "zeros" in all codewords. If during concatenation of codewords the $d = 2$ condition is violated, that is, if the symbol at position 5 of the present codeword and also the symbol at position 1 of the next codeword are "one," the "merging" symbol at position 6 is set to "one" and the symbols at position 1 and 5 are set to "zero." The encoding logic that implements the described merging rule can be very simple. Decoding is done in a similar way.

Rate 8/9, (0, 3) Code

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

$$D^9 = \begin{bmatrix} 208 & 108 & 56 & 29 \\ 193 & 100 & 52 & 27 \\ 164 & 85 & 44 & 23 \\ 108 & 56 & 29 & 15 \end{bmatrix}. \tag{28}$$

The existence of a fixed-length code can be determined with Franaszek's algorithm. It is verified without much difficulty that σ_1 and σ_2 are the principal states (σ_3 and σ_4 are deleted). From any principal state there are at least $293 > 256$ sequences available. This, in principle, concludes the discussion. Patel [23] showed, however, that we can do slightly better. By judiciously discarding a number of potential codewords he arrived 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 (see [24], volume 2), 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.

Eight-to-Fourteen Modulation (EFM) Code

The designers of the Compact Disc [4] system chose, after ample experimental evidence, an RLL code with minimum runlength $d = 2$. The code operates under a second constraint, namely that the dc-content or low-frequency content of the coded bit stream should be as small as possible. The reason for this is that the servo systems used for track following and focusing [25] are controlled by the low-frequency components of the signal read from the disc, and the signal could therefore interfere with the servo systems. 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. The design team opted for a fixed-length block structure. A source block length $m = 8$ is an adequate choice, since the entire format (16-bit audio samples, error correction, etc.) is byte oriented. Table 6 reveals that a fixed-length block code with the parameters $R = 8/16$, $d = 2$ and $k = 10$ is feasible. Specifically, from Table 9 we conclude that a code based on Construction 2 with fourteen information and two two merging bits is possible. In order to reduce the complexity of the decoder logic that transforms the fourteen channel bits into eight data bits, the relationship between data patterns and code patterns have to be optimized. The codebook was compiled with the aid of computer 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 Compact Disc player, the EFM conversion is performed with a programmed logic array of approximately 50 logic functions. Part of the EFM coding table is presented in Table 11,

Table 11 Part of the EFM Coding Table

Data	*Code*	*Data*	*Code*
100	01000100100010	114	10010010000010
101	00000000100010	115	00100000100010
102	01000000100010	116	01000010000010
103	00100100100010	117	00000010000010
104	01001001000010	118	00010001000010
105	10000001000010	119	00100001000010
106	10010001000010	120	01001000000010
107	10001001000010	121	00001001001000
108	01000001000010	122	10010000000010
109	00000001000010	123	10001000000010
110	00010001000010	124	01000000000010
111	00100001000010	125	00001000000010
112	10000000100010	126	00010000100010
113	10000010000010	127	00100000000010

which shows the decimal representation of the 8-bit source word (left column) and its 14-bit channel representation (right column). Space limitations prohibit the presentation of the complete table, for full details the reader is advised to consult the patent literature [26].

The merging bits are primarily intended to ensure that the runlength conditions continue to be satisfied when the codewords are cascaded. 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, two merging bits would be sufficient for continuing to satisfy the runlength conditions. A third merging bit is necessary, however, to provide sufficient freedom for effective suppression of low-frequency content, even though it entails a loss of six percent of the playing time. The three merging bits are redundant, and they are removed from the bit stream in the demodulator.

Figure 5 illustrates, finally, how the merging bits are determined. Our measure of the low-frequency content is

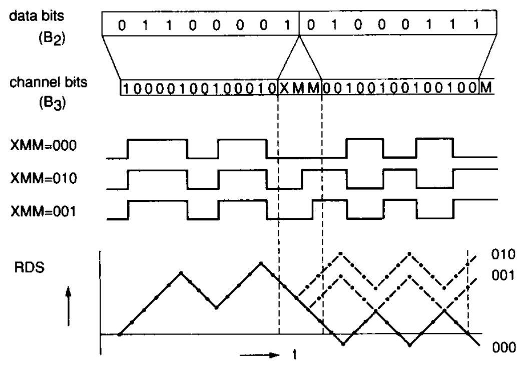


Fig. 5. Strategy for minimizing the running digital sum (RDS). Eight user bits B_2 are translated into fourteen channel bits B_3, the fourteen bits are merged by means of three merging bits in such a way that the runlength conditions continue to be satisfied. The condition that there should be at least two "zeros" between "ones" requires a "zero" at the first merging bit position. In this case there are thus three alternatives for the merging bits: "000," "010," and "001." The encoder chooses the alternative that gives the lowest absolute value of the RDS at the end of a new codeword, i.e., "000" in this case.

the *running digital sum* (RDS); this is the difference between the totals of "zeros" and "ones" accumulated from the beginning of the disc. At the top are shown two 8-bit data words of B_2 and their translation from the codebook into channel symbols (B_3). From the d constraint, 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 xmm = 000, 010, and 001 would give rise to the patterns 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 codeword 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 word ahead, since minimization of the RDS in the short term does not always contribute to longer-term minimization [27]. This more sophisticated algorithm is not implemented in the present equipment.

F. Variable-Length Synchronous Codes

As we have learned 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. The basis of variable-length synchronous codes was laid by Franaszek with his pioneering work reported in [9]. 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. A number of 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 $2n$. To apply the recursive procedure, a basic codeword length n and source word length m are chosen, along with a maximum length Mn. Words may be of length jn, $j = 1, 2, \cdots, M$. The routine involves operations on powers of the D matrix. Two excellent representatives of variable-length codes, to be discussed in the next case studies, are due to Franaszek.

Example 5: Choose the same runlength parameters as in Example 3, namely $d = 2$ and $k = \infty$. After a tedious process of elimination of sequence states, the details of which can be found in [9], a code can be constructed as shown in Table 12.

Table 12 Variable-Length Synchronous (2, ∞) Code

Data		*Code*
0	← →	00
10	← →	0100
11	← →	1000

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 variable-length block code is a set of $\{\mathbf{c}_0, \ldots, \mathbf{c}_{M-1}\}$ of M binary strings. If the codeword $\mathbf{c}_u$ is not the beginning of $\mathbf{c}_v$ 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) may be replaced by one with only three words. □

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 13(a) discloses the code

Table 13a Variable-Length Synchronous (2, 7) Code.

Data		*Code*
10	← →	1000
11	← →	0100
011	← →	000100
010	← →	001000
000	← →	100100
0011	← →	00100100
0010	← →	00001000

table of the rate 1/2, (2, 7) code, which constitutes the bedrock of the IBM3370 and 3380 high-performance rigid disk files [3].

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, (first input bit to the left) then after the appropriate parsing, we obtain

$$in\text{: } 010\ 11\ 10\ 10 \cdots,$$

which, using Table 13(a), is transformed into the corresponding output sequence

$$out\text{: } 001000\ 0100\ 1000\ 1000 \cdots$$

The companion Table 13(b) shows the same codewords and a permutation of the codeword assignments (there are 24

Table 13b Variable-Length Synchronous (2, 7) Code

Data		*Code*
10	← →	0100
11	← →	1000
011	← →	001000
010	← →	100100
000	← →	000100
0011	← →	00001000
0010	← →	00100100

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.

The code conforms to the prefix condition and is therefore self-punctuating. In the case of Table 13(b), 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 the Boolean expression [2]

$$d_0 = x_4\bar{x}_1 + x_7\bar{x}_5x_3 + x_3x_8 + x_6, \tag{29}$$

where d_0 is the decoded symbol and x_i, $1 \le i \le 8$, denote the contents of the shift register. As can be seen from the above Boolean expression, the extension of errors in the decoded data due to single channel symbol errors is limited: any error in a received channel bit may entail a decoding error in up to two subsequent decoded 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 13(a) as originally presented by Franaszek [28] has the drawback that it needs a shift register of length twelve, which increases error extension 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 extension characteristic of the code. How the assignments should be chosen in order to minimize error extension effects is up till now an unsolved problem. Inspection of Table 6 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.

G. Look-Ahead Encoding Technique

Another class of design techniques documented in the literature [29], [22], [30], [31], [18], [32] 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. The code design is guided by the approximate eigenvector inequality. Let the code rate be $m/n < C(d, k)$, where m and n, are positive integers. As said before, an approximate eigenvector $\mathbf{v}$ is a nonnegative integer vector, in this context not necessarily two-valued, satisfying

$$D^n\mathbf{v} \ge 2^m\mathbf{v}. \tag{30}$$

If the matrix D^n does not have a submatrix with row sums at least 2^m, then some component $\mathbf{v}$ will be larger than unity and look-ahead is required. In [18] an approach to finding such a vector in practice is given. Whatever method is used, there may be multiple solutions for the vector $\mathbf{v}$. The choice of the vector may affect the complexity of the code so constructed. The largest component of $\mathbf{v}$ determines the maximum look-ahead span, and the presence in $\mathbf{v}$ of any entries that are not powers of two complicates the coding rules. 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, 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 (1, 7) are available in various forms. Jacoby and Kost [32] described a 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 14(a). 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 14a Basic Coding Table (1, 7) Code

Data	*Code*
00	101
01	100
10	001
11	010

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 substitutions in order to eliminate successive "ones." The process of substitutions in these four combinations is revealed in Table 14(b). The encoding function can be expressed in the form of Boolean equations. In decoding the codewords, those cases in which substitution is made during encoding can be treated without ambiguity, because all three bits of the succeeding codeword are simultaneously "zero."

Table 14b Substituting Coding Table (1, 7) Code

Data	*Code*
00.00	101.000
00.01	100.000
10.00	001.000
10.01	010.000

H. Sliding Block Code Algorithm

The sliding block coding algorithm of Adler, Coppersmith, and Hassner [33] evolved from a field of abstract mathematics known as *symbolic dynamics*. The key idea in this construction method is that one modifies the finite-state transition machine by splitting and merging some of the channel states to obtain a new finite-state machine. An example, taken from [34], may serve to illustrate the idea.

Example 6: We construct a rate 2/3, (0, 1) code. The connection matrices D and D^3 are

$$D = \begin{bmatrix} 1 & 1 \\ 1 & 0 \end{bmatrix} \quad D^3 = \begin{bmatrix} 3 & 2 \\ 2 & 1 \end{bmatrix}. \tag{31}$$

The finite-state transition diagrams associated with D and D^3 are shown in Fig. 6. An eigenvector inequality is given by

$$D^3\mathbf{v} = \begin{bmatrix} 3 & 2 \\ 2 & 1 \end{bmatrix}\begin{bmatrix} 2 \\ 1 \end{bmatrix} \ge 2^2\begin{bmatrix} 2 \\ 1 \end{bmatrix} = 2^2\mathbf{v}. \tag{32}$$

The approximate eigenvector $\mathbf{v} = (2, 1)$ indicates that state 1 will be split into two states, while state 2 will not be split.

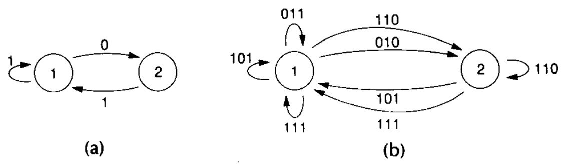


Fig. 6. (a) Finite-state transition diagram of (0,1) sequence. (b) Finite-state transition diagram of third extension of (0,1) sequence. After Siegel 1985 [34].

The two states into which state 1 is split are called 1^1 and 1^2. The outgoing edges of state 1 are partitioned into two groups which are assigned to the two 'offspring' states. All edges which entered state 1 are redirected to both offspring states in the split finite-state diagram. The splitting rule requires that the sum of the weights, where the *weight* of a state is defined as the value of the corresponding component of $\mathbf{v}$, of the terminal states of edges in a group must be an integer multiple of the approximate eigenvalue, 2^2, with the possible exception of one group. We split the edges into groups (011, 110, 010) and (101, 111), both of which have total weight 4. The resultant diagram is shown in Fig. 7. It

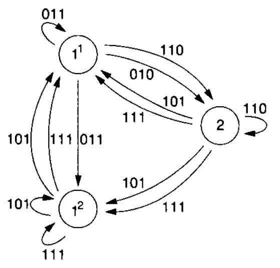


Fig. 7. Split graph. After Siegel 1985 [34].

generates the same set of strings as D^3, but has at least four outgoing edges from each state. By discarding and merging of states we obtain a finite-state machine representation of the encoder (see Fig. 8).

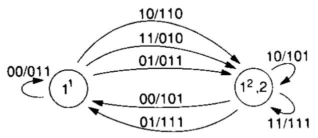


Fig. 8. Finite-state machine representing a (0,1) code. After Siegel 1985 [34].

Decoding can be established with a shift register of length six, and therefore the error extension effect is confined to at most four data bits. □

VII. Conclusions and Remarks

This survey has presented a description of a number of properties of run-length-limited sequences. In the last part of this paper, we have provided the reader with a large assortment of construction techniques. *A priori*, it is hard 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, code designs based on $(dklr)$ sequences may offer a more attractive solution. 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.

References

[1] S. M. C. Borgers, W. A. L. Heijnemans, E. de Niet, and P. H. N. de With, "An experimental digital VCR with 40 mm drum, single actuator and DCT-based bit-rate reduction," *IEEE Trans. Consumer Electr.*, vol. CE-34, pp. 597-605, Aug. 1988.

[2] J. S. Eggenberger and P. Hodges, "Sequential encoding and decoding of variable word length, fixed rate data codes," US Patent 4,115,768, 1978.

[3] T. D. Howell, "Analysis of correctable errors in the IBM 3380 disk file," *IBM J. Res. Develop.*, vol. 28, no. 2, pp. 206-211, March 1984.

[4] J. P. J. Heemskerk and K. A. S. Immink, "Compact disc: System aspects and modulation," *Philips Techn. Review*, vol. 40, no. 6, pp. 157-164, 1982.

[5] C. V. Freiman and A. D. Wyner, "Optimum block codes for noiseless input restricted channels," *Information and Control*, vol. 7, pp. 398-415, 1964.

[6] W. H. Kautz, "Fibonacci codes for synchronization control," *IEEE Trans. Inform. Theory*, vol. IT-11, pp. 284-292, 1965.

[7] A. Gabor, "Adaptive coding for self-clocking recording," *IEEE Trans. Electronic Computers*, vol. EC-16, pp. 866-868, Dec. 1967.

[8] D. T. Tang and L. R. Bahl, "Block codes for a class of constrained noiseless channels," *Information and Control*, vol. 17, pp. 436-461, 1970.

[9] P. A. Franaszek, "Sequence-state encoding for digital transmission," *Bell Syst. Tech. J.*, vol. 47, pp. 143-157, Jan. 1968.

[10] N. N. Vorobév, *Fibonacci Numbers.* Gainsville New Classics Library, 1983.

[11] C. E. Shannon, "A mathematical theory of communication," *Bell Syst. Techn. J.*, vol. 27, pp. 379-423, July 1948.

[12] A. Gallopoulos, C. Heegard, and P. H. Siegel, "The power spectrum of run-length-limited codes," *IEEE Trans. Commun.*, vol. COM-37, pp. 906-917, Sept. 1989.

[13] T. D. Howell, "Statistical properties of selected recording codes," *IBM J. Res. Develop.*, vol. 33, no. 1, pp. 60-73, Jan. 1989.

[14] K. A. S. Immink, "Some statistical properties of maxentropic runlength-limited sequences," *Philips J. Res.*, vol. 38, pp. 138-149, 1983.

[15] E. Zehavi and J. K. Wolf, "On runlength codes," *IEEE Trans. Inform. Theory*, IT-34, pp. 45-54, Jan. 1988.

[16] L. J. Fredrickson and J. K. Wolf, "Coding using multiple block (*d*, *k*) codes," Proceedings *IEEE Conf. on Communications*, pp. 1623-1627, June 1989.

[17] P. A. Franaszek, "On synchronous variable-length coding for discrete noiseless channels," *Information and Control*, vol. 15, pp. 155-164, 1969.

[18] A. Lempel and M. Cohn, "Look-ahead coding for input-restricted channels," *IEEE Trans. Inform. Theory*, vol. IT-28, pp. 933-937, Nov. 1982.

[19] R. S. Varga, *Matrix Iterative Analysis.* Prentice-Hall, Inc., Englewood Cliffs, 1962.

[20] P. A. Franaszek, "Sequence-state methods for run-length-limited coding," *IBM J. Res. Develop.*, vol. 14, pp. 376-383, July 1970.

[21] G. F. M. Beenker and K. A. S. Immink, "A generalized method for encoding and decoding runlength-limited binary sequences," *IEEE Trans. Inform. Theory*, vol. IT-29, pp. 751-754, Sept. 1983.

[22] G. V. Jacoby, "A new look-ahead code for increasing data density," *IEEE Trans. Magn.*, vol. MAG-13, pp. 1202-1204, no. 5, Sept. 1977. See also patent GB 1590404 June 1981.

[23] A. M. Patel, "Improved encoder and decoder for a byte-oriented rate 8/9, (0, 3) code," *IBM Techn. Disclosure Bull.*, vol. 28, pp. 1938, 1985.
[24] C. D. Mee and E. D. Daniel, *Magnetic Recording.* McGraw-Hill Book Company, New York, 1987.
[25] M. G. Carasso, J. B. H. Peek, and J. P. Sinjou, "The compact disc digital audio system," *Philips Techn. Review*, vol. 40, no. 6, pp. 151–156, 1982.
[26] K. A. S. Immink and H. Ogawa, "Method for encoding binary data," US Patent 4,501,000, Feb. 1985.
[27] K. A. S. Immink and U. Gross, "Optimization of low-frequency properties of eight-to-fourteen modulation," *Proc. Fourth Int. Conf. Video and Data Recording*, Southampton, pp. 375–384, 1982, also in *The Radio and Electronic Engineer*, vol. 53, no. 2, pp. 63–66, Feb. 1983.
[28] P. A. Franaszek, "Run-length-limited variable length coding with error propagation limitation," US Patent 3,689,899, Sept. 1972.
[29] A. M. Patel, "Zero-modulation encoding in magnetic recording," *IBM J. Res. Develop.*, vol. 19, pp. 366–378, July 1975.
[30] P. A. Franaszek, "Construction of bounded delay codes for discrete noiseless channels," *IBM J. Res. Develop.*, vol. 26, pp. 506–514, 1982.
[31] M. Cohn and G. V. Jacoby, "Run-length reduction of 3PM code via look-ahead technique," *IEEE Trans. Magn.*, vol. MAG-18, pp. 1253–1255, Nov. 1982.
[32] G. V. Jacoby and R. Kost, "Binary two-thirds rate code with full word look-ahead," *IEEE Trans. Magn.*, vol. MAG-20, pp. 709–714, Sept. 1984. See also M. Cohn, G. V. Jacoby, and C. A. Bates, US Patent 4,337,458, June 1982.
[33] R. L. Adler, D. Coppersmith, and M. Hassner, "Algorithms for sliding block codes. An application of symbolic dynamics to information theory," *IEEE Trans. Inform. Theory*, vol. IT-29, pp. 5–22, Jan. 1983.
[34] P. H. Siegel, "Recording codes for digital magnetic storage," *IEEE Trans. Magn.*, vol. MAG-21, pp. 1344–1349, no. 5, Sept. 1985.

Bibliography and Survey Papers

[35] G. Bouwhuis, J. Braat, A. Huijser, J. Pasman, G. van Rosmalen, and K. Schouhamer Immink, *Principles of Optical Disc Systems.* Adam Higler Ltd., Bristol and Boston, 1985.
[36] J. Isailovic, *Videodisc and Optical Memory Systems.* Prentice-Hall, Inc., Englewood Cliffs, New Jersey, 1985.
[37] J. Watkinson, *The Art of Digital Audio.* Focal Press, London, 1988.
[38] K. W. Cattermole, *Principles of Pulse Code Modulation.* Iliffe Books Ltd, London, 1969.
[39] K. W. Cattermole and J. J. O'Reilly, *Problems of Randomness in Communication Engineering*, vol. 2, Pentech Press, London, 1984.
[40] K. W. Cattermole, "Principles of digital line coding," *Int. J. Electron.*, vol. 55, pp. 3–33, July 1983.
[41] F. Jorgensen, *The Complete Handbook of Magnetic Recording.* TAB Books, Blue Ridge Summit, Pennsylvania, July 1980.
[42] H. Kobayashi, "A survey of coding schemes for transmission or recording of digital data," *IEEE Trans. Commun. Techn.*, vol. COM-19, pp. 1087–1099, Dec. 1971.
[43] N. D. Mackintosh, "The choice of a recording code," *Proc. Int. Conf. on Video and Data Recording*, IERE Conf. Proc. 43, Southampton, pp. 77–120, July 1979.
[44] R. W. Wood, "Magnetic recording systems," *Proc. IEEE*, vol. 74, pp. 1557–1569, Nov. 1986.
[45] R. E. Blahut, *Principles and Practice of Information Theory.* Addison Wesley, 1987.
[46] K. A. S. Immink, *Coding Techniques for Digital Recorders.* Prentice-Hall, Inc., Englewood Cliffs, New Jersey, 1990.

Kees A. Schouhamer Immink (Fellow, IEEE) was born in Rotterdam, The Netherlands, on December 18, 1946. He received the B.S. degree from the Rotterdam Polytechnic in 1967, and the M.S. and Ph.D. degrees from the Eindhoven University of Technology in 1974 and 1985, respectively, all in electrical engineering.

He joined the Philips Research Laboratories, Eindhoven, in 1968. His work first involved the signal processing side of optical recording systems, and he later became responsible for the design and development of channel coding techniques for the Compact Disc, Compact Disc Video, and experimental erasable optical audio discs. In 1986, he was appointed senior scientist of the Philips Research magnetic recording group. He is the author of numerous papers in the field of coding techniques for optical and magnetic recorders, of the book *Coding Techniques for Digital Recorders*, and is a co-author of the book *Principles of Optical Disk Systems*. He holds more than thirty patents, mainly in the area of optical recording.

Dr. Immink was awarded a Fellowship by the Audio Engineering Society in 1985 for "his work in the area of optical laser disc and for detailed study of channel codes for the Compact Disc." He is also a Fellow of the Institution of Electrical Engineers (London).