

# Prefix-Synchronized Run-Length-Limited Sequences

Kees A. S. Immink, *Fellow, IEEE*, and Henk D. L. Hollmann

**Abstract**—Run-length-limited sequences, or  $(d, k)$ -sequences, are used in recording systems to increase the efficiency of the channel. Applications are found 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. In digital recorders, the coded information is commonly grouped in large blocks, called *frames*. Each frame starts with a synchronization pattern, or marker, used by the receiver to identify the frame boundaries and to synchronize the decoder. In most applications, the sync pattern follows the prescribed  $(d, k)$  constraints, and it is *unique*; that is, in the encoded sequence, no block of information will agree with the sync pattern except those specifically transmitted at the beginnings of the frames. Usually, the above arrangement is termed *prefix-synchronized format*. The prefix-synchronized format imposes an extra constraint on the encoded sequences, and will therefore result in a loss of capacity. There is an obvious loss in capacity resulting from the fact that the sender is periodically transmitting sync patterns, and there is a further loss in capacity since the sender is not permitted to transmit patterns equal to the sync pattern during normal transmission. The relative reduction in channel capacity associated with the imposition of this additional channel constraint is calculated. Examples of  $(d, k)$  codes that permit the prefix-synchronization format, based for the purpose of illustration on the sliding-block coding algorithm, are presented.

## I. INTRODUCTION

SINCE the early 1970's, coding methods based on  $(d, k)$ -constrained sequences have been widely used in high-capacity storage systems as magnetic and optical disks or tapes. Properties and applications of  $(d, k)$ -constrained sequences, or run-length-limited sequences as they are often called, are surveyed in [1]. A binary sequence is said to be  $(d, k)$  constrained if the number of "zeros" between pairs of consecutive "ones" lies between  $d$  and  $k$ ,  $k > d$ . An encoder has the task of translating arbitrary user information into a constrained channel sequence. Commonly, the user data is partitioned into words of length  $m$ , and under the coding rules, these  $m$ -tuples are translated into  $n$ -tuples, called codewords. A certain number of codewords constitute a *frame*. All frames are marked with a unique sequence of symbols, referred to as a *marker* or *synchronization pattern*, in short

sync pattern. The frames are chosen so as to assure the nonoccurrence of this marker except when specifically transmitted at the beginnings of each frame. A receiver restores the channel bit clock—usually a phase locked loop is used for this purpose—and subsequently the synchronization pattern is used to identify the frame and codeword boundaries. The sync pattern and the related synchronization circuitry must be designed with great care, as the effect of sync slippage is devastating since entire blocks of information might be lost, which is just as bad as a massive dropout.

Prefix-synchronization codes can be designed by judiciously discarding a number of potential codewords in such a way that the given marker is not a codeword and also does not occur anywhere in the coded sequence as juxtapositions of codewords or the given marker. Alternatively, potential sync patterns can be found by making use of "byproducts" of code implementation. Since the capacity of  $(d, k)$ -constrained sequences is irrational (see Ashley and Siegel [2]), it is clear that code implementations which, by necessity, operate at rates of the form  $m/n$ , where  $m$  and  $n$  are integers, can never attain 100% of the capacity. It was noted by Howell [3] that implemented codes differ from maxentropic, "ideal," sequences by the addition of certain constraints, which he called *incidental constraints*. He found that certain bit patterns, which could readily be used as a basis for constructing sync patterns, are absent in sequences generated by the popular  $(1, 7)$  and  $(2, 7)$  codes.

The main objections one may have to this simple approach are, first, that the sync pattern is found in a rather heuristic and incidental fashion and, second, we have (generally speaking) no idea how far we are away from a sound engineering compromise between redundancy and efficiency and specifically how the choice of a certain sync pattern may affect the complexity of the encoder and decoder. Given that both the choice of the  $(d, k)$  parameters and the choice of the sync pattern involve a careful evaluation, it is clearly appropriate to explore approaches and strategies of a more general, rather than specific, applicability in which the issues of sync pattern selection and code design are beneficially combined.

In 1960, frame synchronization of unconstrained binary sequences in a general setting was addressed by Gilbert [4] (see also Stiffler [5], where a detailed description is given of the synchronization issue). Gilbert showed, among other things, that to each given frame length, there corresponds some optimum length of the sync pattern. Gilbert's work was extended, in 1978, by Guibas and Od-

Manuscript received September 1990; revised August 12, 1991. This work was presented in part at the IEEE 1991 International Symposium on Information Theory, Budapest, Hungary, June 24–28, 1991, and at the 12th Information Theory Symposium in the Benelux, Veldhoven, The Netherlands, May 1991.

The authors are with Philips Research Laboratories, 5600 JA Eindhoven, The Netherlands.

IEEE Log Number 9104247.

lyzko [6], who provided elegant combinational arguments to establish closed-form expressions for generating functions. In this paper, we will concentrate on the frame synchronization problem of  $(d, k)$  sequences. We commence, in Section II with a brief description of  $(d, k)$ -constrained sequences, and proceed with the examination of the channel capacity. It will be shown that for certain sync patterns, called *repetitive-free* sync patterns, the capacity can be formulated in a simple manner as it is solely a function of the  $(d, k)$  parameters and the length of the sync pattern. For each forbidden pattern and  $(d, k)$  constraints, methods for enumerating constrained sequences are given. In the final sections, we deal with design considerations of schemes for encoding and decoding. Examples of prefix-synchronized  $(d, k)$  codes, based for the purpose of illustration on the sliding-block coding algorithm, are presented.

## II. PRELIMINARIES

Sequences (with a leading "one") that meet prescribed  $(d, k)$  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  $10^j$ , where  $0^j$  stands for a sequence of  $j$  consecutive "zeros." The sync pattern is composed of  $p$  phrases,  $T_{s_1}, T_{s_2}, \dots, T_{s_p}$ , and since it is assumed that the sync pattern obeys the prescribed  $(d, k)$  constraints, we have  $s_i \in dk$ , where  $dk = \{d + 1, \dots, k + 1\}$ . The  $p$ -tuple  $s = (s_1, \dots, s_p)$  is used as a shorthand notation to describe the sync pattern. For example,  $s = (2, 1, 3)$  refers to the sync pattern "101100." To keep notation to a minimum,  $s$  will represent both the  $p$ -tuple  $(s_1, \dots, s_p)$  and the string  $10^{s_1-1} \dots 10^{s_p-1}$ ; whichever one is referred to should be clear from the context. The length of the sync pattern,  $L(s)$ , is defined by:

$$L(s) = \sum_{i=1}^p s_i. \quad (1)$$

It should be appreciated that, as a result of the  $(d, k)$  constraints in force, the sync pattern is preceded by at least  $d$  "zeros," and followed by a "one" and at least  $d$  "zeros." (Of course, unless  $s_p = k + 1$ , the "one" starting the phrase following the sync pattern must be a part of the binary pattern used by the synchronization circuitry at the receiver.) It is, therefore, a matter of debate to say that the sync pattern length is  $L(s)$  or  $L(s) + 2d + 1$ . The adopted definition (1) is slightly more convenient, as we will see shortly.

Each frame of coded information consists of the prescribed sync pattern  $s$  and a string of  $l$  cascaded phrases  $T_{i_1}, \dots, T_{i_l}$ . The frame is organized according to the format

$$(T_{s_1}, T_{s_2}, \dots, T_{s_p}, T_{i_1}, T_{i_2}, \dots, T_{i_l}). \quad (2)$$

The total number of binary digits contained in a frame is prescribed and is denoted by  $L_{\text{frame}}$ ; that is,

$$L_{\text{frame}} = \sum_{j=1}^p s_j + \sum_{j=1}^l i_j. \quad (3)$$

The valid choices of  $T_{i_1}, \dots, T_{i_l}$  are those for which no  $p$  consecutive phrases taken from

$$(T_{s_2}, T_{s_3}, \dots, T_{s_p}, T_{i_1}, T_{i_2}, \dots, T_{i_l}, T_{s_1}, T_{s_2}, \dots, T_{s_{p-1}}) \quad (4)$$

agree with the sync pattern. Dictionaries satisfying this constraint are called *prefix-synchronized run-length-limited codes*. It is relevant to enumerate the number of distinct  $L_{\text{frame}}$ -tuples given the above conditions. Following this enumeration problem, we will deal with a related problem, namely the computation of the number of constrained sequences when the frame length  $L_{\text{frame}}$  is allowed to become very large.

## III. ENUMERATION OF SEQUENCES

In this section, we will address the problem of counting the number of constrained sequences. Our methods can be viewed as an extension of those of Guibas and Odlyzko [6] but can, in fact, be traced back to the work of Schuetzenberger on semaphore codes [7, remark 3], see also Berstel and Perrin [8, sect. II-7 and notes following sec. VI]. Schuetzenberger attributes these results to Von Mises and Feller [9].

First we develop some notation. Let  $F$  be the collection of all sequences  $T$  of the form

$$T = (T_{i_1}, \dots, T_{i_n}), \quad i_j \in dk, \quad j = 1, \dots, n \quad (5)$$

composed of  $n > p$  phrases of length

$$L(T) = \sum_{j=1}^n i_j, \quad (6)$$

with the properties that

$$\begin{aligned} \text{F1): } & (i_1, \dots, i_p) \\ & = (i_{n-p+1}, \dots, i_n) = (s_1, \dots, s_p) \end{aligned} \quad (7)$$

$$\begin{aligned} \text{F2): } & (i_j, \dots, i_{j+p-1}) \\ & \neq (s_1, \dots, s_p), \quad j = 2, \dots, n - p. \end{aligned} \quad (8)$$

With the collection  $F$ , we associate the generating function  $f(z)$  defined by:

$$f(z) = \sum_{T \in F} z^{-L(T)}. \quad (9)$$

We denote by  $f_N$  the coefficient of  $z^{-N}$  in  $f(z)$ . Our aim will be to enumerate the number of distinct binary  $N$ -tuples in  $F$ , i.e., to determine the numbers  $f_N$ . Observe that the number  $f_{L_{\text{frame}} + L(s)}$  is nothing but the number of admissible frames. Following [6], we introduce two additional collections of sequences  $G$  and  $H$ , defined as follows. The collection  $G$  consists of all sequences  $T$  as in (5) composed of  $n \geq p$  phrases such that:

$$\text{G1): } (i_1, \dots, i_p) = (s_1, \dots, s_p) \quad (10)$$

$$\begin{aligned} \text{G2): } & (i_j, \dots, i_{j+p-1}) \\ & \neq (s_1, \dots, s_p), \quad j = 2, \dots, n - p + 1 \end{aligned} \quad (11)$$

and  $H$  consists of all sequences  $T$  as in (5) composed of  $n \geq 0$  phrases such that

$$\begin{aligned} \text{H1): } & (i_j, \dots, i_{j+p-1}) \\ & \neq (s_1, \dots, s_p), \quad j = 1, \dots, n-p+1. \end{aligned} \quad (12)$$

Note that by definition the empty sequence  $\Lambda$  is contained in  $H$ ,  $s \in G$ , and the three collections  $F$ ,  $G$ , and  $H$  are mutually disjoint. With these collections  $G$  and  $H$ , we associate generating functions  $g(z)$  and  $h(z)$  defined as

$$g(z) = \sum_{T \in G} z^{-L(T)}; \quad h(z) = \sum_{T \in H} z^{-L(T)}. \quad (13)$$

We now proceed to determine  $f(z)$ ,  $g(z)$ , and  $h(z)$ . The idea is to derive relations between these generating functions from certain combinatorial relations between the collections  $F$ ,  $G$ , and  $H$ . To start with, we observe the following. Let  $T = (T_{i_1}, \dots, T_{i_n})$  be a sequence in  $G$ . Then the sequence  $T * (T_i) = (T_{i_1}, \dots, T_{i_n}, T_i)$ ,  $i \in dk$ , is contained in  $F$  or in  $G \setminus \{s\}$ , but not in both since  $F$  and  $G$  are disjoint. On the other hand, if  $T = (T_{i_1}, \dots, T_{i_n})$ ,  $n \geq p+1$ , is contained in  $F$  or in  $G \setminus \{s\}$ , then the sequence  $(T_{i_1}, \dots, T_{i_{n-1}})$  is contained in  $G$ . We conclude that there is a one-to-one correspondence between sequences

$$T * (T_i), \quad T \in G, \quad i \in dk$$

and sequences in

$$F \cup G \setminus \{s\}.$$

From this observation, the following lemma is immediate.

*Lemma 1:*

$$g(z) P_{dk}(z) = f(z) + g(z) - z^{-L(s)}$$

where

$$P_{dk}(z) = \sum_{i \in dk} z^{-L(T_i)} = \sum_{i \in dk} z^{-i}. \quad (14)$$

*Proof:* We have

$$\begin{aligned} g(z) P_{dk}(z) &= \sum_{T \in G} z^{-L(T)} \cdot \sum_{i \in dk} z^{-L(T_i)} \\ &= \sum_{T \in G} \sum_{i \in dk} z^{-L(T * (T_i))} \\ &= \sum_{\hat{T} \in F \cup G, \hat{T} \neq s} z^{-L(\hat{T})} \\ &= f(z) + g(z) - z^{-L(s)}. \quad \square \end{aligned}$$

Similarly, we may derive a one-to-one correspondence between sequences

$$(T_i) * T, \quad T \in H, \quad i \in dk$$

and sequences in

$$H \cup G \setminus \{\Lambda\}$$

(recall that  $G$  and  $H$  are disjoint), which leads to the next lemma.

*Lemma 2:*

$$P_{dk}(z) h(z) = h(z) + g(z) - 1.$$

*Proof:* Similar to the proof of Lemma 1. (Note that  $z^{-L(\Lambda)} = z^0 = 1$ .)  $\square$

Before we can write down the last relationship, we require some definitions.

Define the  $(p-i)$ -tuples  $\mathbf{h}^{(i)} = (s_1, \dots, s_{p-i})$  and  $\mathbf{t}^{(i)} = (s_{i+1}, \dots, s_p)$ , so  $\mathbf{h}^{(i)}$  and  $\mathbf{t}^{(i)}$  consist of the  $p-i$ ,  $i = 0, \dots, p-1$ , first and last phrases of the marker  $s$ , respectively. Let  $L(\mathbf{h}^{(i)})$  be the length of the first  $p-i$  phrases of  $s$  or

$$L(\mathbf{h}^{(i)}) = \sum_{j=1}^{p-i} s_j. \quad (15)$$

The *autocorrelation function* of  $s$ , denoted by  $r$ , is a binary vector of length  $p$  which is defined by  $r_i = 0$  if  $\mathbf{h}^{(i)} \neq \mathbf{t}^{(i)}$  and  $r_i = 1$  if  $\mathbf{h}^{(i)} = \mathbf{t}^{(i)}$ . Obviously,  $r_0 = 1$ . An example may serve to explain why  $r$  is termed autocorrelation function. Let  $s = (1, 2, 1, 2, 1)$ , then Fig. 1 exemplifies the process of forming the autocorrelation function. If a marker tail coincides with a marker head, we have  $r_i = 1$ ; else  $r_i = 0$ .

If  $r_i = 0$ ,  $1 \leq i \leq p-1$ , that is, if no proper tail equals a proper head of the marker, we say that the marker is *repetitive free*.

We are now in the position to prepare Lemma 3. To that end, let  $i$ ,  $0 \leq i \leq p-1$ , be such that  $r_i = 1$ , and let  $T = (T_{i_1}, \dots, T_{i_n}) \in G$ . By definition of  $G$ ,  $i_j = s_j$  for  $j = 1, \dots, p$ . Since  $r_i = 1$ , we have

$$\begin{aligned} (T_{s_1}, \dots, T_{s_i}) * T \\ &= (T_{s_1}, \dots, T_{s_i}, T_{s_1}, \dots, T_{s_p}, T_{i_p+1}, \dots, T_{i_n}) \\ &= (T_{s_1}, \dots, T_{s_i}, T_{s_i+1}, \dots, T_{s_p}, T_{s_p-i+1}, \dots, \\ &\quad T_{s_p}, T_{i_p+1}, \dots, T_{i_n}) \\ &= s * \hat{T}, \end{aligned}$$

where  $\hat{T} = (T_{s_p-i+1}, \dots, T_{s_p}, T_{i_p+1}, \dots, T_{i_n})$ . Moreover,  $\hat{T}$  is a proper suffix of  $T$ , hence  $\hat{T} \in H$ .

Conversely, let  $\hat{T} = (T_{i_1}, \dots, T_{i_n}) \in H$ . Consider the word  $U = s * \hat{T} = (U_{i_1}, \dots, U_{i_n+p})$ . Let  $j$  be the largest number such that  $(U_{i_j}, \dots, U_{i_j+p-1}) = s$ . Note that, since  $\hat{T} \in H$ ,  $1 \leq j \leq p$  holds. From the way in which  $j$  was defined, it follows that  $T := (U_{i_j}, \dots, U_{i_n+p}) \in G$  and moreover that  $i := j-1$  satisfies  $r_i = 1$ .

From the above, we conclude that there exists a one-to-one correspondence between sequences

$$(T_{s_1}, \dots, T_{s_i}) * T, \quad 0 \leq i \leq p-1, \quad r_i = 1, \quad T \in G$$

and sequences

$$s * \hat{T}, \quad \hat{T} \in H.$$

From this observation, the following lemma easily follows.

*Lemma 3:*

$$(1 + F_r(z))g(z) = z^{-L(s)}h(z)$$

$i$	1	2	1	2	1	$r_i$
1		1	2	1	2	0
2			1	2	1	1
3				1	2	0
4					1	1

Fig. 1. Process of forming the autocorrelation function  $r$ .

where the polynomial  $F_r(z)$  is

$$F_r(z) = \sum_{i=1}^{p-1} r_i z^{L(\mathbf{h}^{(i)}) - L(s)}. \quad (16)$$

*Proof:* If  $r_i = 1$ , then  $\mathbf{h}^{(i)} = \mathbf{t}^{(i)} = (s_{i+1}, \dots, s_p)$ , whence

$$L(\mathbf{h}^{(i)}) - L(s) = -L(s_1, \dots, s_i). \quad (17)$$

Note also that

$$r_0 z^{L(\mathbf{h}^{(0)}) - L(s)} = 1. \quad (18)$$

Therefore,

$$\begin{aligned} (1 + F_r(z))g(z) &= \left( \sum_{0 \leq i \leq p-1} z^{-L(s_1, \dots, s_i)} \right) \sum_{T \in G} z^{-L(T)} \\ &= \sum_{\substack{0 \leq i \leq p-1 \\ r_i=1}} \sum_{T \in G} z^{-L(s_1, \dots, s_i) * T} \\ &= \sum_{\hat{T} \in H} z^{-L(s * \hat{T})} \\ &= z^{-L(s)} h(z). \quad \square \end{aligned}$$

From the relations between  $f(z)$ ,  $g(z)$ , and  $h(z)$  as described in Lemmas 1–3, we may derive the following result.

*Theorem 1:*

$$z^{L(s)} f(z) = \frac{F_r(z)(P_{dk}(z) - 1) - z^{-L(s)}}{P_{dk}(z) - 1 - z^{-L(s)} - (1 - P_{dk}(z))F_r(z)}. \quad (19)$$

*Proof:* From Lemmas 2 and 3, an expression for  $g(z)$  not involving  $h(z)$  may be derived. If this expression for  $g(z)$  is combined with Lemma 1, the theorem follows easily.  $\square$

*Corollary 1:* The number of admissible frames of length  $N = L_{\text{frame}}$  is the coefficient of  $z^{-N}$  in the power series expression of the RHS of (19).  $\square$

It is immediate from Theorem 1 that the number of constrained sequences is solely a function of the marker length  $L(s)$  if the marker is repetitive free. Although Theorem 1 is useful for enumerating the number of constrained sequences, it shows its greatest utility in investigating the asymptotic behavior of the number of constrained sequences when the sequence length is allowed to become very large. This asymptotic behavior is directly related to the (noiseless) capacity to be discussed in the ensuing section.

### A. Capacity

The number of distinct  $(d, k)$  sequences of length  $n$ , denoted by  $N_{dk}(n)$ , grows, for sufficiently large  $n$ , exponentially:

$$N_{dk}(n) \approx a_1 2^{C(d,k)n} \quad (20)$$

where  $a_1$  is a constant independent of  $n$ , and the rate of growth  $C(d, k)$  is termed the (noiseless) capacity of the  $(d, k)$ -constrained channel. The capacity of a  $(d, k)$ -constrained channel, denoted by  $C(d, k)$ , is the base-2 logarithm of the largest real root of the characteristic equation [10]

$$P_{dk}(z) = \sum_{i \in dk} z^{-i} = 1. \quad (21)$$

For the  $d$ -constrained case,  $k = \infty$ , we have the characteristic equation

$$P_{d,\infty}(z) = \sum_{i=d+1}^{\infty} z^{-i} = \frac{z^{-d}}{z-1} = 1. \quad (22)$$

Using the above characteristic equations, it is not difficult to compute the capacity of  $(d, k)$ -constrained channels. The results of the computations have been tabulated in [1].

The capacity  $C(d, k, s)$  of  $(d, k)$  sequences where the marker  $s$  is forbidden can be found from the next theorem.

*Theorem 2:*  $C(d, k, s) = \log_2 \lambda$ , where  $\lambda$  is the largest real root of

$$P_{dk}(z) - z^{-L(s)} - (1 - P_{dk}(z))F_r(z) = 1. \quad (23)$$

*Proof:* Follows from Theorem 1. Note that the numerator and denominator of the RHS of (19) have no common factors.  $\square$

An upper and lower bound to the capacity  $C(d, k, s)$  are given in the next corollary.

*Corollary 2:* For given sync pattern length  $L(s)$ ,

$$\log_2 \lambda_l \leq C(d, k, s) \leq \log_2 \lambda_u$$

where  $\lambda_l$  is the largest real root of

$$P_{dk}(z) - z^{-L(s)} = 1$$

and  $\lambda_u$  is the largest real root of

$$P_{dk}(z) - z^{-L(s)} - (1 - P_{dk}(z)) \sum_{i=1}^{p-1} z^{-i(d+1)} = 1.$$

The lower bound is attained if  $s$  is repetitive free and the upper bound is attained if  $s$  is of the form  $(d+1, \dots, d+1)$ .

*Proof:* Let

$$q(z) := P_{dk}(z) - 1 - z^{-L(s)} - (1 - P_{dk}(z))F_r(z)$$

$$a(z) := P_{dk}(z) - 1 - z^{-L(s)}$$

and

$$b(z) := P_{dk}(z) - 1 - z^{-L(s)} - (1 - P_{dk}(z)) \sum_{i=1}^{p-1} z^{-i(d+1)}.$$

We start with the lower bound. Since  $a(\lambda_l) = 0$ , we have

$$q(\lambda_l) = -(1 - P_{dk}(\lambda_l))F_r(\lambda_l) = \lambda_l^{-L(s)} F_r(\lambda_l) \geq 0.$$

TABLE I  
CAPACITY OF THE (1, 3)-CONSTRAINED CHANNEL

$s$	$C(1, 3, s)$	$\Sigma v_i$	$\max \{v_i\}$
1000	0.40569	—	—
1010	0.46496	—	—
10100	0.45316	—	—
100010	0.48673	—	—
100100	0.50630	18	4
101010	0.51737	30	6
1000100	0.50902	11	3
1001010	0.50902	20	4
1010010	0.51606	33	6
10001000	0.52934	11	3
10001010	0.52352	12	3
10010010	0.52352	18	4
10010100	0.52785	26	4
10100010	0.52669	32	6
10101010	0.53691	30	6

Since  $q(z) = -1$  for  $z \rightarrow \infty$ , and since  $\lambda$  is the largest real zero of  $q(z)$ , this implies  $\lambda_l \leq \lambda$ . Equality holds if  $r = (1, 0, \dots, 0)$ , that is, if the sync pattern  $s$  is repetitive free.

The upper bound follows from a similar argument. Since  $1 \leq \lambda_l \leq \lambda$ , we have  $\lambda^{-1} \leq 1$ . Moreover,  $L(s_1, \dots, s_i) \geq i(d+1)$ , hence

$$F_r(\lambda) \leq \sum_{i=1}^{p-1} \lambda^{-L(s_1, \dots, s_i)} \leq \sum_{i=1}^{p-1} \lambda^{-i(d+1)}.$$

From  $q(\lambda) = 0$ , it follows that

$$(1 - P_{dk}(\lambda))(1 + F_r(\lambda)) = -\lambda^{-L(s)} < 0$$

whence  $1 - P_{dk}(\lambda) < 0$ . Therefore,

$$b(\lambda) = (1 - P_{dk}(\lambda))(F_r(\lambda) - \sum_{i=1}^{p-1} \lambda^{-i(d+1)}) \geq 0.$$

Since  $b(z) = -1$  for  $z \rightarrow \infty$ , and since  $\lambda_u$  is the largest real zero of  $b(z)$ , this implies  $\lambda \leq \lambda_u$ .  $\square$

Tables I-III give  $C(d, k, s)$  for selected  $(d, k)$  parameters. (The right-hand columns are related to the code design to be discussed in a later section.) From Table I we observe, for example, that  $s = (3, 3)$  is the shortest sync pattern that admits of a code with rate 1:2, while the shortest repetitive-free sync patterns that admit of the same code rate are  $s = (4, 3)$  and  $(3, 2, 2)$  (and, of course, their reversed counter parts, which are not in the table). In the Appendix, an analysis is given to approximate  $C(d, k, s)$ . In the next section, we will discuss what the consequences are in terms of encoder and decoder complexity.

#### IV. CODE DESIGN

In this section, we use the preceding theory to provide some examples of code design based on the sliding-block code algorithm [11]. The starting point of the sliding-block code algorithm is a description of the channel constraints by a labeled directed graph. An  $N_s$ -state (labeled) graph is characterized by an  $N_s \times N_s$  adjacency matrix  $D$  with nonnegative integer entries and a labeling output function  $\zeta$  that maps the transitions (edges) into the chan-

TABLE II  
CAPACITY OF THE (1, 7)-CONSTRAINED CHANNEL

$s$	$C(1, 7, s_i)$	$\Sigma v_i$	$\max \{v_i\}$
1000000	0.66262	—	—
1000010	0.66262	—	—
1000100	0.66262	—	—
1001010	0.66262	—	—
1010010	0.66422	—	—
10000000	0.66903	99	19
10000010	0.66903	129	19
10000100	0.66903	147	19
10001000	0.67047	164	19
10001010	0.66903	160	19
10010010	0.66903	275	30
10010100	0.66998	252	27
10100010	0.66965	279	27
10101010	0.67302	142	15
100000010	0.67296	65	11
100000100	0.67296	76	11
100001000	0.67296	88	11
100001010	0.67296	86	11
100010010	0.67296	102	11
100010100	0.67296	102	11
100100010	0.67296	156	16
100100100	0.67448	153	16
100101010	0.67296	152	16
101000010	0.67320	198	19
101001010	0.67320	202	19

TABLE III  
CAPACITY OF THE (2, 7)-CONSTRAINED CHANNEL

$s$	$C(2, 7, s)$	$\Sigma v_i$	$\max \{v_i\}$
1000000	0.48913	—	—
1000100	0.48913	—	—
10000000	0.49791	—	—
10000100	0.49791	—	—
10001000	0.50202	160	20
100000100	0.50395	67	10
100001000	0.53095	108	14
100100100	0.50841	124	14
1000000100	0.50811	61	10
1000001000	0.50811	70	10
1000010000	0.50952	80	10
1000100100	0.50811	85	10
1001000100	0.50887	134	14
10000000100	0.51097	23	4
10000001000	0.51097	39	6
10000010000	0.51097	62	8
10000100100	0.51097	79	10
10001000100	0.51097	96	10
10001001000	0.51148	86	10
10010000100	0.51134	138	14

nel alphabet. In the next section, we will describe the graph that represents the  $(d, k)$  and synchronization constraints. Thereafter, as a straightforward application, we present some examples of code design.

#### A. Graph Description

We will now proceed to show that the constraints can be represented by an  $(L + k + 1 - s_1)$ -state graph. Admissible sequences emitted by the graph exclude the sync pattern as well as those sequences which violate the  $(d, k)$  constraints. Let the  $(L + k + 1 - s_1)$  states be denoted by  $\sigma_1, \dots, \sigma_L, \hat{\sigma}_1, \dots, \hat{\sigma}_{k+1-s_1}$ . The graph

occupies state  $\sigma_i$ ,  $1 \leq i \leq L$ , if it is possible to form a marker by extending the generated sequence with  $L - i + 1$  symbols, or stated alternatively, if the  $i$  most recently emitted symbols equal the  $i$  first symbols of the marker (and if  $i$  is maximal with this property). The graph is in the state  $\hat{\sigma}_i$ ,  $1 \leq i \leq k + 1 - s_1$ , if, to form a marker, the generated sequence must be extended with  $L + 1$  symbols and if the most recently emitted "zero" run length is  $s_1 - 1 + i$ . The structure of the graph can be described as follows. Let  $\mathbf{m} = (m_1, \dots, m_p)$  denote the  $p$ -tuple formed by the  $p$  most recent phrases generated by the graph. Evidently,  $m_i \in dk$ ,  $i \leq p - 1$  and  $m_p \in \{1, \dots, k + 1\}$ . Let  $l$  be the largest integer,  $1 \leq l \leq p$  ( $s_1, \dots, s_{l-1} = (m_{p-l+1}, \dots, m_{p-1})$  and  $m_p \leq s_l$  hold. (The first condition is considered to be vacuously true if  $l = 1$ .) If no such integer can be found, then set  $l = 0$ . Then, if  $l > 0$ , the graph is in the state

$$\sigma_j, j = \sum_{i=p-l+1}^p m_i$$

else it is in the state  $\hat{\sigma}_j$ ,  $j = m_p - s_1$ . The various transitions between the states and the output labeling function are implied by the definition of the states and the constraints in force. Fig. 2, for example, depicts an 11-state graph that represents the (2, 7) constraints and does not generate the marker  $s = (5, 3)$ .

### B. Results

The graph described in the previous section serves as the input to the sliding-block algorithm. The sliding-block code design is guided by the *approximate eigenvector inequality*. Let the code rate be  $m/n < C(d, k)$ , where  $m$  and  $n$ ,  $m < n$ , are positive integers. An approximate eigenvector  $\mathbf{v}$  is a nonnegative integer vector satisfying

$$D^n \mathbf{v} \geq 2^m \mathbf{v}. \quad (24)$$

If  $\lambda$  is the largest positive eigenvalue of the adjacency matrix  $D$ , then by the Perron-Frobenius theory (see e.g., Varga [12]) there exists a vector  $\mathbf{v}$  whose entries  $v_i$  are nonnegative integers satisfying (24), where  $n/m \leq \log_2 \lambda$ . The following algorithm, taken from Franaszek [13], (see also Adler *et al.* [11]) is an approach to finding such a vector.

Choose an initial vector  $\mathbf{v}^{(0)}$  whose entries are  $v_i^{(0)} = \beta$ , where  $\beta$  is a nonnegative integer. Define inductively

$$v_i^{(u+1)} = \min \left( v_i^{(u)}, \text{entier} \left( 2^{-m} \sum_{j=1}^{N_i} [D]_{ij}^n v_j^{(u)} \right) \right) \quad (25)$$

where  $\text{entier}(x)$  denotes the largest integer less than or equal  $x$ . Let

$$\mathbf{v} \equiv \mathbf{v}^{(u)}$$

where  $u$  is the first integer such that  $\mathbf{v}^{(u+1)} = \mathbf{v}^{(u)}$ . There are two situations: a)  $\mathbf{v} > 0$  and b)  $\mathbf{v} = 0$ . Case a) means that we have found an approximate eigenvector, and in case b) there is no solution, so we increase  $\beta$  and start from the top again. There may be multiple solutions for

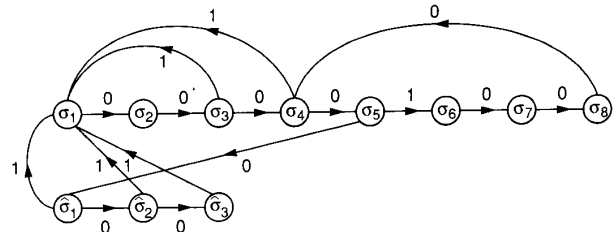


Fig. 2. Eleven-state source that describes the (2, 7) constraints and does not generate the pattern "10000100(1)." Note that the last "one," written in parentheses, indicates the start of a new phrase.

the vector  $\mathbf{v}$ . The choice of the vector may affect some significant parameters of the code so constructed. The largest component of  $\mathbf{v}$  is related to the error propagation in the decoding process, and the parameter  $\Sigma v_i$  is a factor that has an influence on the encoder complexity. After finding an approximate eigenvector using (25), it is sometimes possible to find a better eigenvector by a systematic trial and error method. Adler *et al.* [11] suggested reducing one of the components of  $\mathbf{v}$  and applying (25) again.

At least in principle, it is now possible to run a computer program in order to evaluate the suitability of a certain sync pattern and to assess the concomitant complexity of the encoder and decoder hardware. When the sync pattern is long, however, the evaluation of each valid sync pattern quite easily becomes an engineering impossibility. We opted for a slightly different approach. A rough estimate of the encoder and decoder complexity can be based on two simple indicators: a)  $\Sigma v_i$  and b)  $\max \{v_i\}$ . It has been shown by Adler *et al.* [11] that the size of the encoder is upper bounded by  $\Sigma v_i$  and the error propagation properties are related to the number of rounds in the splitting process which is, in turn, connected to  $\max \{v_i\}$ . Results of computations are collected in Tables I-III for rate-1/2, (1, 3), rate-2/3, (1, 7), and rate-1/2, (2, 7) codes, respectively. In general terms, the figures reflect our intuition that as the capacity of the constrained channel increases, it becomes easier to implement the code. Some outcomes, however, are intriguing as they contradict this general rule. Specifically, we observe that both parameters  $\Sigma v_i$  and  $\max \{v_i\}$  suggest that the rate 1/2, (1, 3) code with sync pattern (3, 3) is easier to implement than the same code with the sync pattern (2, 3, 2), while one might expect otherwise as the capacity of the latter is significantly greater. It is not at all obvious whether this is an artifact of the indicators or whether other parameters play a role. Further study, for example using the tools presented by Marcus and Roth [14], is needed to answer these questions. In the next section, we will describe a worked example of the design of a rate 1/2, (1, 3) code.

### C. Worked Example

We consider the design of a rate 1/2, (1, 3) code which does not generate the marker  $s = (4, 3)$  or "1000100(1)." The capacity of the channel is (see Table I)  $C(1, 3, s = (4, 3)) = 0.50902$ , so that this code achieves an efficiency

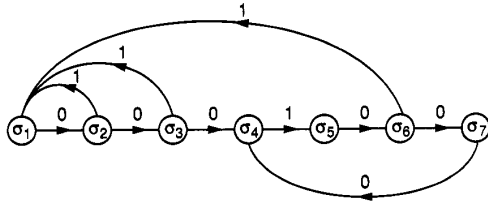


Fig. 3. Seven-state source that describes the (1, 3) constraints and does not generate a pattern "1000100(1)."

of  $0.5/0.50902 = 98.2\%$ . Fig. 3 shows the graphical representation of a seven-state source whose output sequences obey the given constraints.

The adjacency matrix  $D$  associated with this source is

$$D = \begin{bmatrix} 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 1 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 \end{bmatrix}. \quad (26)$$

Invoking (25), we find the approximate eigenvector  $\mathbf{v}^T = (2, 3, 2, 1, 1, 2, 0)$ . After the previous spadework, it is now a simple matter to assemble, using the sliding-block coding algorithm, a rate  $1/2$ , (1, 3) sliding-block encoder. The encoder is defined by two characterizing functions: the next-state function  $g(\sigma_i, \bar{\beta}_u)$  and the output function  $h(\sigma_i, \bar{\beta}_u)$ , where  $\bar{\beta}_u$ ,  $u = 1, \dots, 2^m$ , are the  $2^m$  source words, and  $\sigma_i$ ,  $i = 1, \dots, N_e$ , are the  $N_e$  encoder states. The output and next-state function are shown in Table IV.

The implementation of the rate  $1/2$ , (1, 3) encoder is elementary; the encoder comprises a three-stage register to store the actual state (three bits are sufficient to represent the five encoder states), and a  $(1 + 3) \rightarrow (2 + 3)$  logic array for looking up the 2-b codeword and the 3-b next state. Decoding can be accomplished by a sliding-window decoder with a window length of eight channel bits, thus limiting error propagation to at most four decoded user bits. No attempt has been made to improve this parameter.

#### D. Incidental Constraints

Run-length-limited codes operate at rates of the form  $m/n$ , where  $m$  and  $n$  are integers. It was shown by Ashley and Siegel [2] that, save a trivial exception, the capacity of  $(d, k)$ -constrained sequences is irrational. It is, therefore, clear that code implementations can never attain 100% of the capacity. It was noted by Howell [3] that implemented codes differ from maxentropic, "ideal," sequences by the addition of a few constraints, which he

TABLE IV  
CODEBOOK OF RATE  $1/2$ , (1, 3) CODE

$\sigma_i$	$g(\sigma_i, 0)$	$h(\sigma_i, 0)$	$g(\sigma_i, 1)$	$h(\sigma_i, 1)$
1	1	10	4	10
2	1	00	5	00
3	2	01	3	01
4	1	00	3	10
5	3	01	3	10

called incidental constraints. Howell [3] described in detail the incidental constraints of three popular  $(d, k)$  codes, namely the rate  $2/3$ , (1, 7) code [15], the rate  $1/2$ , (2, 7) code [16], and the rate  $1/2$ , (1, 3) code. He found that certain bit patterns do not occur in sequences generated by these codes. The codes above exhibit other incidental constraints. They are, however, not relevant for synchronization purposes and are therefore not discussed here. An important beneficial consequence of the presence of incidental constraints is that prefix synchronization can often be established by exploiting the incidental constraints so that information capacity is not wastefully dissipated. It is an instructive exercise to compare the three above codes, where we could base the sync pattern on the incidental constraints, with codes found with the more direct approach outlined in the theory developed.

1) *Rate  $1/2$ , (1, 3) Code:* 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 state-dependent block code with codewords of length  $n = 2$ . The encoding rules underlying the MFM code are shown in Table V. 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 MFM sequences have a minimum and maximum run length of  $d = 1$  and  $k = 3$ , respectively.

These rules are easily translated into a two-state encoder. Decoding of the MFM code is simply accomplished by discarding the redundant first bit in each received 2-b block. It is not difficult to check that, under MFM rules, the output sequence "10 00" is not valid but "01 00 01" is valid, so that "1000" cannot serve as a sync pattern. The shortest sequence that does not occur in an MFM sequence is the 12-b sequence "100010010001." Perusal of Table I reveals that the shortest markers giving room for the design of a rate  $1/2$  code are of length six. The rate  $1/2$ , (1, 3) code described in Section IV-C has the advantage that its sync word is 8-b long instead of twelve bits required in the MFM code. On the other hand, the MFM code has a simpler implementation and the decoder window length is shorter, namely two bits, and is thus more favorable in terms of error propagation.

*Rate  $1/2$ , (2, 7) Code:* There are a great many kinds of rate- $1/2$  codes known for the parameters  $d = 2$  and

TABLE V  
 CODING RULES MFM CODE

Source	Output
0	x0
1	01

$k = 7$ . We will concentrate on the variable-length variant invented by Franaszek [16] and modified by Eggenberger and Hodges [17]. The code can be encoded with a six-state encoder (Howell [3] showed how to devise a five-state encoder) and decoded with a sliding-block decoder of length eight. Howell [3] found that this (2, 7) code does not produce 12-b sequences of the form  $10^7 10^2 1$ , so that the sequence  $s = (8, 3)$  can be used as a marker in this scheme. In Table III, we observe that there are nine candidate sync patterns  $s$  of length  $L(s) < 11$  which permit the construction of a rate  $1/2$  code; the shortest marker has length eight. Rate- $1/2$  codes which embed each of these candidate markers were generated using the sliding-block coding algorithm. Approximately 10 rounds of splitting steps are required for the codes with the markers of length 8 and 9; the corresponding encoders utilize approximately 30 states. The codes for the markers of length 10 require 8 splitting rounds. The simplest code found uses the marker  $s = (7, 3)$ ; the code requires 23 encoder states and can be decoded with a decoder window of 23 bits.

3) *Rate 2/3, (1, 7) Code:* The rate  $2/3$ , (1, 7) code described by Jacoby and Kost [15] is a lookahead code. It can be encoded by a five-state encoder and decoded by a decoder with a window of seven bits. Howell [3] has observed that the set of sequences generated by this code is invariant under time reversal. The 16-b pattern  $10^6 10^7 1$  and its reverse  $10^7 10^6 1$  do not occur in sequences generated by the (1, 7) code. We have evaluated a number of shorter sync patterns, but could not find a code that rivaled the Jacoby/Kost code in complexity.

## V. CONCLUSIONS

We have investigated the prefix synchronization of run-length-limited sequences. The capacity of  $(d, k)$ -constrained channels giving room for the prefix-synchronization format has been analyzed. It has been shown that for certain sync patterns, termed repetitive-free sync patterns, the capacity can be formulated in a simple manner, as it is solely a function of the  $(d, k)$  parameters and the length of the sync pattern. For presentation purposes, the complexity consequences of short sync patterns have been investigated for the rate- $1/2$  codes with parameters (1, 3) and (2, 7), and for the rate  $2/3$ , (1, 7) code. We have not found a code that rivaled (1, 7) and (2, 7) codes whose sync patterns are based on their respective incidental constraints. A worked example of a rate  $1/2$ , (1, 3) code giving room for the inclusion of a unique sync pattern of length seven has been presented.

## APPENDIX

### APPROXIMATIONS TO THE CAPACITY BOUNDS

In this Appendix, we derive approximations to the lower and upper bound to the capacity  $C(d, k, s)$ . Corollary 2 asserts that for a given marker length  $L(s)$ ,  $\log_2 \lambda_l \leq C(d, k, s) \leq \log_2 \lambda_u$ , where  $\lambda_l$  is the largest real root of

$$P_{dk}(z) - z^{-L(s)} = 1$$

and  $\lambda_u$  is the largest real root of

$$P_{dk}(z) - z^{-L(s)} - (1 - P_{dk}(z)) \sum_{i=1}^{p-1} z^{-i(d+1)} = 1.$$

We commence with the lower bound. To that end, let  $\mu$  be the largest real root of the characteristic equation

$$P_{dk}(z) = \sum_{i \in dk} z^{-i} = 1 \quad (27)$$

and let  $\lambda_l = \mu + \Delta \lambda_l$  be the largest real root of

$$P_{dk}(z) - z^{-L(s)} = 1. \quad (28)$$

Then

$$P_{dk}(\mu + \Delta \lambda_l) - (\mu + \Delta \lambda_l)^{-L(s)} = 1.$$

Differentiating and working out yields

$$\Delta \lambda_l \approx -\frac{\mu}{\bar{T}} \mu^{-L(s)}, \quad L(s) \gg 1$$

where the average run length  $\bar{T}$  is

$$\bar{T} = \sum_{i \in dk} i \mu^{-i}.$$

Thus,

$$\begin{aligned} C(d, k, s) &> \log_2 (\mu + \Delta \lambda_l) \\ &= C(d, k) + \log_2 \left( 1 + \frac{\Delta \lambda_l}{\mu} \right) \\ &\approx C(d, k) - \frac{1}{\ln(2) \bar{T}} \mu^{-L(s)}. \end{aligned} \quad (29)$$

In a similar fashion, we find for the upper bound  $\lambda_u = \mu + \Delta \lambda_u$ :

$$\Delta \lambda_u \approx -\frac{\mu}{\bar{T}} \frac{\mu^{-L(s)}}{1 + \sum_{i=1}^{p-1} \mu^{-i(d+1)}}.$$

We have

$$1 + \sum_{i=1}^{p-1} \mu^{-i(d+1)} \approx \frac{1}{1 - \mu^{-(d+1)}}, \quad L(s) \gg 1$$

and from (22), we infer that, for  $k \gg 1$ ,

$$\mu^{-(d+1)} \approx 1 - \mu^{-1}$$

so that

$$\Delta \lambda_u \approx -\frac{1}{\bar{T}} \mu^{-L(s)}, \quad L(s) \gg 1, k \gg 1.$$

Thus, for the relative capacity loss we find the following approximations:

$$\frac{1}{\ln(2)T} \mu^{-L(s)-1} < C(d, k) - C(d, k, s) < \frac{1}{\ln(2)T} \mu^{-L(s)}, \quad L(s) \gg 1. \quad (30)$$

#### REFERENCES

- [1] K. A. S. Immink, "Runlength-limited sequences," *Proc. IEEE*, vol. 78, no. 11, pp. 1745-1759, Nov. 1990.
- [2] J. Ashley and P. H. Siegel, "A note on the Shannon capacity of runlength-limited codes," *IEEE Trans. Inform. Theory*, vol. IT-33, pp. 601-605, July 1987.
- [3] T. D. Howell, "Statistical properties of selected recording codes," *IBM J. Res. Develop.*, vol. 33, no. 1, pp. 60-73, Jan. 1989.
- [4] E. N. Gilbert, "Synchronization of binary messages," *IEEE Trans. Inform. Theory*, vol. IT-6, pp. 470-477, Sept. 1960.
- [5] J. J. Stiffler, *Theory of Synchronous Communications*. Englewood Cliffs, NJ: Prentice-Hall, 1971.
- [6] L. J. Guibas and A. M. Odlyzko, "Maximal prefix-synchronized codes," *SIAM J. Appl. Math.*, vol. 35, no. 2, pp. 401-418, 1978.
- [7] M. P. Schuetzenberger, "On the synchronizing properties of certain prefix codes," *Inform. Contr.*, vol. 7, pp. 23-36, 1964.
- [8] J. Berstel and D. Perrin, *Theory of Codes*. Orlando, FL: Academic, 1985.
- [9] W. Feller, *An Introduction to Probability Theory and Its Applications*, Vol. 1. New York: Wiley, 1959.
- [10] C. E. Shannon, "A mathematical theory of communication," *Bell Syst. Tech. J.*, vol. 27, pp. 379-423, July 1948.
- [11] 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.
- [12] R. S. Varga, *Matrix Iterative Analysis*. Englewood Cliffs, NJ: Prentice-Hall, 1962.
- [13] P. A. Franaszek, "Construction of bounded delay codes for discrete noiseless channels," *IBM J. Res. Develop.*, vol. 26, pp. 506-514, 1982.
- [14] B. H. Marcus and R. M. Roth, "Bounds on the number of states in encoder graphs for input-constrained channels," *IEEE Trans. Inform. Theory*, vol. 37, no. 3, pp. 742-758, May 1991.
- [15] G. V. Jacoby and R. Kost, "Binary two-thirds rate code with full word look-ahead," *IEEE Trans. Magnet.*, vol. MAG-20, pp. 709-714, Sept. 1984; also in M. Cohn, G. V. Jacoby, and C. A. Bates, III, U.S. Patent 4 337 458, June 1982.
- [16] P. A. Franaszek, "Run-length-limited variable length coding with error propagation limitation," U.S. Patent 3 689 899, Sept. 1972.
- [17] J. S. Eggenberger and P. Hodges, "Sequential encoding and decoding of variable word length, fixed rate data codes," U.S. Patent 4 115 768, 1978.



**Kees A. S. Immink** (M'81-SM'86-F'90) was born in Rotterdam, The Netherlands, on December 18, 1946. He received the B.S. degree from Rotterdam Polytechnic in 1967, and the M.S. and Ph.D. degrees from the Eindhoven University of Technology, Eindhoven, The Netherlands, in 1974 and 1985, respectively, all in electrical engineering.

He joined Philips Research Laboratories, Eindhoven, The Netherlands, in 1968, where his work involved the signal processing side of optical recording systems.

He later became responsible for the design and development of channel coding techniques for the compact disc, compact disc video, experimental erasable optical audio discs, R-DAT, and DCC recorders. In 1986, he was appointed Senior Scientist of the Philips Research Magnetic Recording Group where his current research focuses on digital magnetic audio and video recorders for consumer applications. He holds more than 30 patents, mainly in the area of optical recording, has written numerous papers in the field of coding techniques for optical and magnetic recorders, and is author of the monograph *Coding Techniques for Digital Recorders* and a coauthor of *Principles of Optical Disc Systems*.

Dr. Immink is a Fellow of the AES.



**Henk D. L. Hollmann** was born in Utrecht, The Netherlands, on March 10, 1954. He received the M.S. degree in mathematics from Eindhoven University of Technology, Eindhoven, The Netherlands.

In 1982 he joined CNET, Issy-les-Moulineaux, France, where he worked mainly on number theoretical transforms. Currently, he is with the Philips Research Laboratory, Eindhoven, The Netherlands. His research interests include discrete mathematics/combinatorics, error-correcting codes, and digital signal processing.