

**Codes for Mass Data Storage
Systems
Second Edition**

**Codes for Mass Data Storage
Systems
Second Edition**

Kees A. Schouhamer Immink

Institute for Experimental Mathematics,
University of Essen-Duisburg, Essen, Germany

Shannon Foundation Publishers, Eindhoven, The Netherlands.

Published by Shannon Foundation Publishers, Eindhoven, The Netherlands.
CIP-gegevens Koninklijke Bibliotheek, Den Haag
Schouhamer Immink, K.A.
ISBN 90-74249-27-2
Information can be obtained from the living web site

www.shannonfoundation.org/book.html

All rights are reserved.

©2004 Shannon Foundation Publishers, Eindhoven, The Netherlands
No part of the material protected by this copyright notice may be reproduced or utilized in any form or by any means, electronic or mechanical, including photocopying, recording or by any other storage and retrieval system, without the written permission from the copyright owner.

Printed in The Netherlands

Preface

Since the early 1980s we have witnessed the digital audio and video revolution: the Compact Disc (CD) has become a commodity audio system. CD-ROM and DVD-ROM have become the *de facto* standard for the storage of large computer programs and files. Growing fast in popularity are the digital audio and video recording systems called DVD and BluRay Disc. The above mass storage products, which form the backbone of modern electronic entertainment industry, would have been impossible without the usage of advanced coding systems.

Pulse Code Modulation (PCM) is a process in which an analogue, audio or video, signal is encoded into a digital bit stream. The analogue signal is sampled, quantized and finally encoded into a bit stream. The origins of digital audio can be traced as far back as 1937, when Alec H. Reeves, a British scientist, invented pulse code modulation [287]. The advantages of digital audio and video recording have been known and appreciated for a long time. The principal advantage that digital implementation confers over analog systems is that in a well-engineered digital recording system the sole significant degradation takes place at the initial digitization, and the quality lasts until the point of ultimate failure. In an analog system, quality is diminished at each stage of signal processing and the number of recording generations is limited. The quality of analog recordings, like the proverbial 'old soldier', just fades away. The advent of ever-cheaper and faster digital circuitry has made feasible the creation of high-end digital video and audio recorders, an impracticable possibility using previous generations of conventional analog hardware.

The general subject of coding for digital recorders is very broad, with its roots deep set in history. In digital recording (and transmission) systems, channel encoding is employed to improve the efficiency and reliability of the channel. Channel coding is commonly accomplished in two successive steps: (a) error-correction code followed by (b) recording (or modulation) code. Error-correction control is realized by adding extra symbols to the conveyed message. These extra symbols make it possible for the receiver to correct errors that may occur in the received message.

In the second coding step, the input data are translated into a sequence with special properties that comply with the given "physical nature" of the

recorder. Of course, it is very difficult to define precisely the area of recording codes and it is even more difficult to be in any sense comprehensive. The special attributes that the recorded sequences should have to render it compatible with the physical characteristics of the available transmission channel are called *channel constraints*. For instance, in optical recording a '1' is recorded as pit and a '0' is recorded as land. For physical reasons, the pits or lands should neither be too long or too short. Thus, one records only those messages that satisfy a *run-length-limited* (RLL) constraint. This requires the construction of a code which translates arbitrary source data into sequences that obey the given constraints. Many commercial recorder products, such as Compact Disc and DVD, use an RLL code.

The main part of this book is concerned with the theoretical and practical aspects of coding techniques intended to improve the reliability and efficiency of mass recording systems as a whole. The successful operation of any recording code is crucially dependent upon specific properties of the various subsystems of the recorder. There are no techniques, other than experimental ones, available to assess the suitability of a specific coding technique. It is therefore not possible to provide a cookbook approach for the selection of the 'best' recording code.

In this book, theory has been blended with practice to show how theoretical principles are applied to design encoders and decoders. The practitioner's view will predominate: we shall not be content with proving that a particular code exists and ignore the practical detail that the decoder complexity is only a billion times more complex than the largest existing computer. The ultimate goal of all work, application, is never once lost from sight. Much effort has been gone into the presentation of advanced topics such as in-depth treatments of code design techniques, hardware consequences, and applications. The list of references (including many US Patents) has been made as complete as possible and suggestions for 'further reading' have been included for those who wish to pursue specific topics in more detail.

The decision to update *Coding Techniques for Digital Recorders*, published by Prentice-Hall (UK) in 1991, was made in Singapore during my stay in the winter of 1998. The principal reason for this decision was that during the last ten years or so, we have witnessed a success story of coding for constrained channels. The topic of this book, once the province of industrial research, has become an active research field in academia as well. During the IEEE International Symposia on Information Theory (ISIT) and the IEEE International Conference on Communications (ICC), for example, there are now usually three sessions entirely devoted to aspects of constrained coding. As a result, very exciting new material, in the form of (conference) articles and theses, has become available, and an update became a necessity.

The author is indebted to the Institute for Experimental Mathematics,

University of Duisburg-Essen, Germany, the Data Storage Institute (DSI) and National University of Singapore (NUS), both in Singapore, and Princeton University, US, for the opportunity offered to write this book. Among the many people who helped me with this project, I like to thank Dr. Ludo Tolhuizen, Philips Research Eindhoven, for reading and providing useful comments and additions to the manuscript.

Preface to the Second Edition

About five years after the publication of the first edition, it was felt that an update of this text would be inescapable as so many relevant publications, including patents and survey papers, have been published. The author's principal aim in writing the second edition is to add the newly published coding methods, and discuss them in the context of the prior art. As a result about 150 new references, including many patents and patent applications, most of them younger than five years old, have been added to the former list of references. Fortunately, the US Patent Office now follows the European Patent Office in publishing a patent application after eighteen months of its first application, and this policy clearly adds to the rapid access to this important part of the technical literature.

I am grateful to many readers who have helped me to correct (clerical) errors in the first edition and also to those who brought new and exciting material to my attention. I have tried to correct every error that I found or was brought to my attention by attentive readers, and seriously tried to avoid introducing new errors in the Second Edition.

China is becoming a major player in the art of constructing, designing, and basic research of electronic storage systems. A Chinese translation of the first edition has been published early 2004. The author is indebted to prof. Xu, Tsinghua University, Beijing, for taking the initiative for this Chinese version, and also to Mr. Zhijun Lei, Tsinghua University, for undertaking the arduous task of translating this book from English to Chinese. Clearly, this translation makes it possible that a billion more people will now have access to it.

Kees A. Schouhamer Immink
Rotterdam, November 2004

Contents

Preface	viii
1 Introduction	1
2 Entropy and Capacity	9
2.1 Introduction	9
2.2 Information content, Entropy	10
2.2.1 Entropy of memoryless sources	10
2.2.2 Markov chains	13
2.2.3 Entropy of Markov information sources	18
2.3 Channel capacity of constrained channels	20
2.3.1 Capacity of Markov information sources	21
2.3.2 Sources with variable-length symbols	24
3 Spectral Analysis	29
3.1 Introduction	29
3.2 Stochastic processes	30
3.2.1 Stationary processes	30
3.2.2 Cyclo-stationary processes	32
3.3 Spectra of Markov sources	34
3.4 Description of encoder models	36
3.5 Spectra of block-coded signals	39
3.5.1 Spectral analysis of Moore-type encoders	41
3.5.2 Spectrum of memoryless block codes	45
4 Runlength-limited Sequences: Theory	51
4.1 Introduction	51
4.2 Counting (dk) sequences	54
4.3 Asymptotic information rate	56
4.3.1 State-transition matrix description	60
4.3.2 Useful properties	62
4.4 Maxentropic RLL sequences	64
4.4.1 Spectrum of maxentropic RLL sequences	66
4.4.2 Comparison of results	68

4.5	Other runlength constraints	71
4.5.1	Prefix-synchronized RLL sequences	71
4.5.2	Asymmetrical runlength constraints	79
4.5.3	MTR constraints	81
4.5.4	RLL sequences with multiple spacings	81
4.5.5	$(O, G/I)$ sequences	84
4.6	Weak constraints	85
4.6.1	Capacity of the weakly dk -constrained channel	85
4.7	Multi-level RLL Sequences	87
4.8	Two-dimensional RLL constraints	90
4.9	Appendix: Computation of the spectrum	93
5	RLL Block Codes	95
5.1	Introduction	95
5.2	RLL (d, k) block codes	96
5.2.1	Modified frequency modulation, MFM	98
5.3	Block codes of minimum length	99
5.3.1	Franaszek's recursive elimination algorithm	100
5.3.2	State-independent decoding	104
5.4	Block codes based on $(dklr)$ sequences	106
5.4.1	Generating functions	107
5.4.2	Constructions 1 and 2	108
5.5	Optimal block codes	111
5.5.1	Set-concatenation	111
5.6	Examples of RLL (d, k) block codes	113
5.6.1	EFM	114
5.6.2	Rate $8/9$, $(0, 3)$ code	114
5.6.3	Other k -constrained codes	115
5.6.4	Sequence replacement technique	118
5.6.5	High-rate (klr) codes with uncoded symbols	119
5.7	Block-decodable RLL Codes	122
5.8	Almost block-decodable codes	126
5.8.1	Three Position Modulation (3PM) code	127
5.8.2	Constructions 3, 4, and 5	128
5.8.3	Generalized construction	131
5.8.4	Results and comparisons	132
5.9	Appendix: Generating functions	134
6	Enumerative coding	137
6.1	Introduction	137
6.2	Basics of enumeration	138
6.3	Enumeration of $(dklr)$ sequences	141
6.3.1	Enumeration using floating-point arithmetic	144

6.3.2	Effects of floating point arithmetic	145
6.3.3	Very long block codes	147
6.3.4	Application to (k) -constrained sequences	150
6.4	Error propagation	152
6.5	Alternatives to standard codes	153
6.5.1	Burst error correction	155
6.6	Appendix: Asymptotics	157
7	Sliding-Block Codes	159
7.1	Introduction	159
7.2	Description of a sliding-block decoder	160
7.3	Variable-length (VL) codes	162
7.3.1	Synchronous variable-length RLL codes	164
7.3.2	Examples of synchronous VL codes	164
7.4	Look-ahead encoding technique	170
7.4.1	Rate 2/3, (1,7) code	171
7.5	Sliding-block algorithm	172
7.5.1	Higher-order edge graphs	174
7.5.2	State splitting	176
7.6	Baldwin codes	181
7.6.1	Encoder description	181
7.7	Immink codes	183
7.7.1	Encoder description, $d=1$ case	183
7.7.2	Encoder description, $d=2$ case	187
7.7.3	Very efficient coding schemes	189
7.8	Discussion	191
8	Dc-balanced Codes	195
8.1	Introduction	195
8.2	Preliminaries	198
8.2.1	Capacity of dc-constrained sequences	200
8.2.2	Spectra of maxentropic sequences	203
8.3	Performance assessment	208
8.4	Simple coding schemes	213
8.4.1	Zero-disparity coding schemes	213
8.4.2	Low-disparity coding schemes	215
8.4.3	Polarity switch method	218
8.5	Computation of the spectrum	219
8.6	Performance appraisal	225
8.6.1	Efficiency of simple codes	229
8.7	High rate dc-balanced codes	231
8.8	Dc-free code with odd codeword length	233
8.8.1	Alternative definition of RDS.	234

8.9	Balancing of codewords	236
8.10	Appendix	239
8.10.1	Computation of the sum variance	239
8.10.2	Computation of the correlation	240
9	Higher-Order Spectral Zeros	243
9.1	Introduction	243
9.2	Preliminaries	244
9.3	Enumeration of sequences	245
9.4	Coding with zero-disparity codewords	247
9.4.1	Spectra of zero-disparity codewords	249
9.5	State-dependent encoding	251
9.6	Higher-order dc-constrained codes	253
10	Guided Scrambling	257
10.1	Introduction	257
10.2	Guided scrambling basics	259
10.2.1	Guided scrambling	260
10.3	Analysis of multi-mode dc-free codes	263
10.3.1	The random drawing model	264
10.3.2	Transition probabilities of the finite-state machine	264
10.3.3	Computational results	266
10.3.4	Alternative metrics	266
10.4	Evaluation using Hadamard Transform	269
10.5	Comparison with alternative methods	270
10.6	Weakly constrained codes	272
10.6.1	Weak $(0, k)$ codes	273
11	Dc-free RLL Codes	277
11.1	Introduction	277
11.2	Capacity of DCRLL codes	277
11.3	Spectrum of maxentropic DCRLL sequences	280
11.4	Examples of DCRLL codes	284
11.4.1	ACH-algorithm-based DCRLL codes	285
11.4.2	Dc-control: data level versus channel level	286
11.4.3	Codes with parity preserving word assignment	288
11.4.4	Zero Modulation	291
11.4.5	Miller-Miller code	292
11.5	EFM revisited	293
11.5.1	EFM	293
11.5.2	EFMPlus	296
11.5.3	Alternatives to EFM schemes	300
11.5.4	Performance of EFM-like coding schemes	302

<i>CONTENTS</i>	xiii
12 Further Reading	305
12.1 Selected literature	306
13 Author's Biography	309
Bibliography	311
Index	341

Chapter 1

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 mass storage systems, such as optical and magnetic recording products, has continued unabated ever since the day when the first mechanical computer memories were introduced. Naturally, technological advances such as improved materials, heads, mechanics, and so on have greatly increased the storage capacity, but state-of-the-art storage densities are also a function of improvements in channel coding, the topic addressed in this text.

To understand how digital data is stored and retrieved from a magnetic medium, we need to review a few (very) basic facts. The magnetic material contained on a magnetic disk or tape can be thought of as being made up of a collection of discrete magnetic particles or domains which can be magnetized by a write head in one of two directions. In present systems, digital information is stored along paths in this magnetic media called *tracks*. In its simplest form we store binary digits on a track by magnetizing these particles or domains in one of two directions. Two (equivalent) conventions have been used to map the stored binary digits to the magnetization along a track. In one convention, called *NRZ*, one direction of magnetization corresponds to a stored '1' and the other direction of magnetization corresponds to a stored '0'. In the other convention, called *NRZI*, a reversal of the direction of magnetization represents a stored '1' and a nonreversal of magnetization represents a stored '0'.

In magnetic recording, the stored binary digits usually are referred to as *channel bits*. In all magnetic storage systems used today, either the magnetic media or the head (or heads) used to write and/or read the data move with respect to each other. If this relative velocity is constant, the constant duration of the bit (in seconds) can be transformed to a constant length of the bit along a track.

As has been observed by many authors, the storing and retrieving of digital information from a storage system is a special case of a digital com-

munications channel. Thus, as information theory provides the theoretical underpinnings for digital communications, this theory applies equally well to digital storage systems. 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. 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 or stored. To a casual observer it may appear that coding systems are a recent phenomenon. This is true in a purely commercial context, but research into coding techniques has its roots in information theory which goes back to the pioneering work of Shannon [296]. Shannon offered the world the so-called 'fundamental theorem of information theory', which states that *it is possible to transmit information through a noisy channel at any speed less than the channel capacity with an arbitrarily small probability of error*. This raises the immediate question: How can the promised theoretical land of arbitrarily small error probability be reached 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'. A *code* is a set of rules for assigning, to a source (or input) sequence, another sequence that is recorded. The aim of this transformation is to improve the reliability and/or efficiency of the recording channel.

In recorder systems, channel encoding is commonly accomplished in two successive steps: (a) error-correction code and (b) recording (or modulation) code. The various coding steps are visualized in Figure 1.1 which shows a block diagram of a possible data storage system of this kind. The decoder, the spitting image, or more accurately the 'egami gnittips', of the encoder, reconstitutes, by using the redundancy present in the retrieved signal, the input sequences as accurately as possible. 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 data storage applications is the family of *Reed-Solomon codes* [340]. The reason for their pre-eminence in mass storage systems is that they can combat combinations of random as well as burst errors. Error-correction or error-detection coding techniques are amply dealt with in many excellent textbooks and are not considered in this book.

The arrangement called *channel* or *recording code*, sometimes referred to as *modulation code* or *constrained code*, on which this book 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 called *con-*

strained sequence, which is suitable for the specific recorder requirements. In transmission over electrical or fiber cables, the technique related to the recording code is termed *line coding*. The term recording code encompasses a large variety of codes, and it is therefore difficult to provide an exact and general definition of such a code. In particular, the seeming lack of distinction between error control coding and recording/line coding may easily lead to confusion in the reader's mind. More confusion is added as there are recording codes with error correcting capabilities, or, if you wish, error correcting codes that satisfy certain channel constraints.

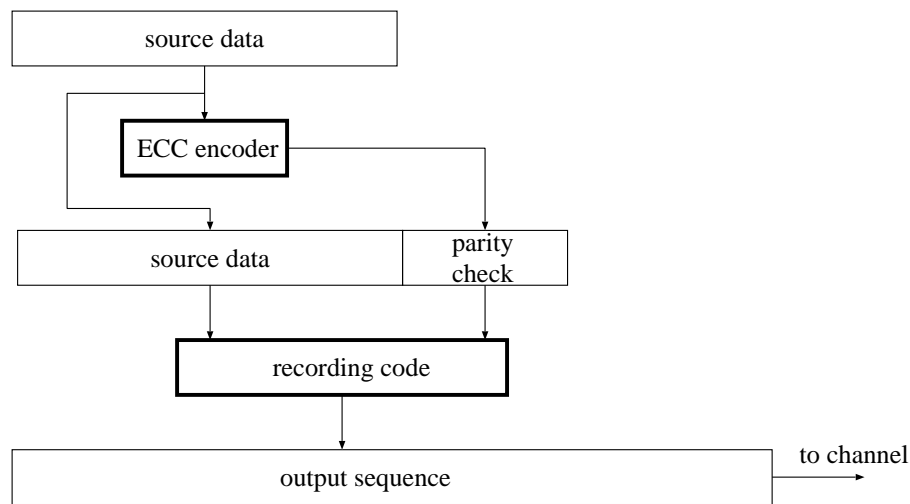


Figure 1.1: Block diagram of the two coding steps in a data storage system. The source data are translated in two successive steps: (a) error-correction code (ECC) and (b) recording (or channel) code. The ECC encoder generates parity check symbols. The source data plus parity check symbols are translated into a constrained output sequence by the recording code. The output sequence is stored on the storage medium in the form of binary physical quantities, for example pits and lands, or positive and negative magnetizations. In this context, the method of putting a second code on top of the first one is usually called a *concatenated scheme*.

An encoder has the task of translating (binary) user data into a sequence that complies with the given channel constraints. On the average, m binary user symbols are translated into n binary channel symbols. Any restriction on the repertoire of symbol sequences diminishes the quantity of information, in bits, to something less than the number of binary digits actually utilized. A measure of the efficiency implied by a particular code is given by the quotient $R = m/n$, where R is called the *information rate* or, in short, rate of the code. The quantity $1 - R$ is called the *redundancy* of the code. Since only a limited repertoire of sequences is used, the rate of a code

that transforms input data into a constrained sequence is by necessity less than unity. The maximum rate given the channel input constraints is often called the *Shannon capacity* or, in short, *capacity* of the input-constrained noiseless channel. A good code embodiment realizes a code rate that is close to the Shannon capacity of the constrained sequences, uses a simple implementation, and avoids the propagation of errors at the decoding process, or, more realistically, one with a compromise among these competing attributes.

Notably spectral shaping and runlength-limited (RLL) codes have found widespread usage in consumer-type mass storage systems such as Compact Disc, DAT, DVD, and so on [150]. Table 1.1 gives a survey of recording codes currently in use by consumer-type mass data storage products.

Table 1.1: Survey of recording codes and their application area

<i>Device</i>	<i>Code</i>	<i>Type</i>	<i>Ref.</i>
Compact Disc	EFM	RLL, dc-free	[130]
MiniDisc	EFM	RLL, dc-free	[354]
DVD	EFMPlus	RLL, dc-free	[151]
BluRay	(1,7)PP	RLL, dc-free	[260]
R-DAT	8-10	dc-free	[259]
floppy and hard disk	(2,7) or (1,7)	RLL	[96]
DCC	ETM	dc-free	[142]
Scoopman	LDM-2	RLL, dc-free	[133]
DVC	24 \rightarrow 25	dc-free, pilot tones	[180]

There are various reasons for the application of a recording code. For example, long sequences of like symbols can easily foil the receiver's electronics such as the timing recovery or the adaptive equalization whose design usually rests on the assumption that the signals are stationary. It seems quite reasonable to try to protect against vexatious waveforms by removing them from the input repertoire. A coding step in which particular sequences are removed to minimize the effect of worst-case patterns is a good example of a recording/line code. The special attributes that the recorded sequences should have to render it compatible with the physical characteristics of the available transmission channel are called *channel constraints*. The channel constraints considered here are deterministic by nature and are always in force. The above case is a typical example of a channel constraint described in time-domain terms: sequences containing runs of like symbols that are too long are forbidden. Channel constraints can also be described in frequency-domain terms. Common magnetic recorders do 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 a coding step.

The two previous examples of recording codes are used to reduce the likelihood of errors by avoiding those pathological sequences which are *a priori* known to be most vulnerable to channel impairment and thus prone to error. *Scrambling*, applied, for example, in the D1 digital video tape recorder [115], is an alternative method often advocated to eliminate 'worst-case' effects in the data. Scramblers use pseudo-random sequences to randomize the statistics of the data, making them look more like a stationary sequence. There are, however, pathological input patterns for which scramblers will fail, since any technique that performs a one-to-one mapping between input and output data and by necessity does not remove error-prone sequences, remains vulnerable to problematic inputs. We can just hope that the probability of occurrence of such error-prone events is reduced.

Recording codes are also used to include position information for servo systems and timing information for clock recovery. A plethora of all kinds of codes are found in this area. They include codes for generating pilot tracking tones and spectral null codes. Specifically in this field, coding by means of a recording code has proven to be a powerful and versatile tool in the hands of the system architect. All the aforementioned coding stages are used in a modern digital video recorder [39].

There are various ways in which digital symbols can be represented by physical quantities. All these involve assigning a range of waveforms of a continuously variable physical function to represent some digital symbol. Most digital data storage systems currently in use are binary and synchronous, which means that in each symbol time interval, or time slot, a condition of, for example, pit or no pit, positive or negative magnetization, etc., is stored. During read-out, the receiver, under the control of the retrieved clock, properly phased with respect to the incoming data, samples the retrieved signal at the middle of each time slot. It is well known that a single pulse transmitted over a bandwidth-limited system is smeared out in time due to the convolution with the channel's impulse response. A sample at the center of a symbol interval is a weighted sum of amplitudes of pulses in several adjacent intervals. This phenomenon is called *intersymbol interference* (ISI). If the product of symbol time interval and system bandwidth is reduced, the ISI will become more severe. A point may eventually be reached at which this effect becomes so severe that the receiver, even in the absence of noise, can no longer distinguish between the symbol value and the ISI and will start to make errors.

Manipulation of the sequence structure with the aim of minimizing the effects of ISI, or to tailor the power spectral density to specific needs, is the domain of the recording, or transmission, codes. We refer to such an operation as (*spectral*) *coding* of the message sequence. As a typical example,

consider the effects of high-pass filtering or a.c. coupling. The a.c. coupling of a channel manifests itself as ISI called *baseline wander*. A common technique to cancel the effects of baseline wander is to deploy a code that has no spectral content at the low-frequency end.

According to Nyquist's classical criteria for distortionless transmission [265] all ISI can be eliminated prior to detection by means of a linear filter which essentially flattens, or equalizes, the characteristics of the channel. In practice there are difficulties in providing correct equalization at all times. Tape surface asperities and substrate irregularities may cause variations or fluctuations in the intimacy of head contact, which changes the response at high frequencies much more than at low frequencies. In disc drives, the varying radius of the tracks results in a linear density variation of about two to one, and the thickness of the air film, which is a function of the disc-head speed, is also subject to change. In disc(k) systems, spindle wobble and disc warp mean that the focal plane is continuously moving and the spot quality will vary due to the finite bandwidth of the focus servo. Optimum equalization is very difficult to maintain under dynamic conditions, although an adaptive equalizer can be used to follow the dynamic changes. An adaptive equalizer has limited accuracy and by necessity is has to react slowly to the variations. A recording code must therefore be designed to show a certain acceptable ruggedness against said dynamic changes of the channel's characteristics that an adaptive or fixed compromise equalizer cannot cope with. This statement applies specifically to the DAT and Compact Disc, which are meant for domestic use and where, therefore, readjustment of the mechanical parameters is out of the question.

The physical parameters mentioned above are associated purely with the one-dimensional communication channel approach. Designing for high linear density (along the recording track) cannot be the sole objective. In a system approach, both dimensions, track and linear density, should be considered. There can be little doubt that the system approach in which the interaction between various subsystems (servos, materials, etc.) is taken into account holds the key to forthcoming significant advances in information density. In magnetic recorders, signal amplitude and signal-to-noise ratio are proportional to the track width. In this sense, linear and track density can be traded against each other. For mechanical design simplicity, high-performance digital tape recorders such as the DAT recorder are designed to operate at relatively high linear densities but low track density. With the use of wide tracks with high linear densities, ISI rather than (additive) noise is frequently the limiting factor. This statement is especially true for optical recording systems, where noise is virtually absent. Unfortunately, it is extremely difficult to analyze or quantify system factors like crosstalk from adjacent tracks, or interaction between data written on the disc or tape with servo systems utilized to follow the tracks. Such charac-

terization methods require much time and effort, as well as the skill of the system engineer who carries out the experiments, if their results are to be consistent. It is conventional to judge the performance of coding schemes in terms of error probability. It should be borne in mind that this is not the sole valid quantity for appraising the performance of a data storage system. It seems to be appropriate to remark that any error correction and detection is totally useless if track loss occurs either through an improper construction of the tracking servo or because of excessive disc or tape damage.

Does this book provide an answer to the question: What code should be employed in a specific data storage system? The short answer to this question is that there is no one answer which is satisfactory under any general set of circumstances. The actual process of selecting and devising a recording code is in fact always a trade-off between various conflicting requirements such as signal-to-noise ratio, clocking accuracy, non-linearities, and inter-symbol interference. 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 the encoding chosen. The problem of code selection is further compounded by non-technical considerations such as the patent situation or familiarity and these factors are to some extent certainly valid. One further, and certainly not the least relevant, factor that affects the choice of a 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.

The difficulty remains, of course, in actually defining the optimal channel constraints and in appraising the code's performance in any specific practical situation. It should be borne in mind that, although our present knowledge of the physics of the recording channel, whether optical or magnetic, is reasonably detailed, there are still large areas where the channel behavior is much more difficult to account for. Theories are bound to fall short of the perfect explanation or prediction because the real world is far too complex to be embraced in a single theory, and because in the real world there are factors at work that are genuinely unpredictable. It is therefore doubtful whether the tenets of information and communication theory can really be of any more help in assessing channel code performance in the physical context of a recording channel. However, information theory affords other tools that can be helpful. The performance of ideal or *maxentropic* constrained sequences, that is, sequences whose information content equals the channel capacity, obviously sets a standard by which the performance of implemented codes can be measured. The attributes of maxentropic constrained sequences provide valuable criteria for assessing both the effects of specific code parameters on the attainable performance, and the effects of important trade-off's which are involved in the system design. The performance measures obtained can serve as a guide. It should be noted, however, that

it can never substitute for experimental results or experience obtained by experiments. Further definitive performance criteria seem to be, at least at present and in this context, beyond the reach of mathematics and definitely belong to the province of the design engineer. It is therefore likely that, in spite of the attractions of the elegant mathematical formulation of information and communication theory, more *ad hoc* approaches to coder/decoder design are here to stay.

Chapter 2

Entropy and Capacity

2.1 Introduction

In this chapter we develop tools for constructing, describing, and analyzing the codes and code properties that appear subsequently in this book. Our main objective in this chapter is to provide an answer to a fundamental question that arises in the analysis of communication systems: Given an information source, how do we quantify the amount of information that the source is emitting?

Generally speaking, information sources are classified in two categories: continuous and discrete. Continuous information sources emit a continuous-amplitude, continuous-time waveform, whereas a discrete information source conveys symbols from a finite set of letters or *alphabet of symbols*. In this book we shall confine ourselves to discrete information sources. Every message emitted by the source contains some information, but some messages convey more information than others. In order to quantify the information content of messages transmitted by information sources, we will study the important notion of a measure of information, called *entropy*, in this chapter. The material presented in this chapter is based on the pioneering and classical work of Shannon [296], who published in 1948 a treatise on the mathematical theory of communications.

A channel that does not permit input sequences containing any of a certain collection of forbidden subsequences is called an *input-restricted* or *constrained channel*. The constraints specify what is possible or allowed and what is not. The physical limitations of, for instance, the time-base regeneration or the adaptive equalization circuitry lead to the engineering conclusion that troublesome sequences that may foil the receiver circuitry have to be discarded, and that therefore not the entire population of possible sequences can be used. To be specific, sequences are permitted that guarantee runs of consecutive 'zero' or 'one' symbols that are neither too short nor too long. This is an example of a channel with runlength constraints (a *run-*

length is the number of times that the same symbol is repeated). Another typical constraint may require that the difference between the numbers of transmitted 'one's and 'zero's is bounded. It should be appreciated that the channel constraints considered here are deterministic by nature and are always in force. The basic notion is that the messages are conveyed over a *noiseless* channel, that is, a channel that never makes errors. Our concern here is to maximize the number of messages that can be sent (or stored) over the channel in a given time (or given area) given the deterministic channel constraints.

The main part of this book is concerned with methods how to transform, that is encode, binary data into a series of binary data that a channel can accept. On the average, m binary user symbols are translated into n binary channel symbols. Obviously, since not the entire population is used, $n \geq m$. A measure of the efficiency implied by a particular code is given by the *code rate*, R , where R is defined as $R = m/n$. The fraction of transmitted symbols that are redundant is $1 - R$. Clearly, an unconstrained channel, that is a channel that permits any arbitrary binary sequence, has unity rate. It is a cardinal question how many of the possible sequences have to be excluded, or, stated alternatively, how much of a rate (loss) one needs to suffer at most to convert arbitrary data into a sequence that satisfies the specified channel constraints. The maximum rate feasible given determinate constraints is often called, in honor of the founder of information theory, the *Shannon capacity*, or *noiseless capacity* of the input-constrained channel.

We commence with a quantitative description of information content of various discrete sources. The central concept of channel capacity is introduced in Section 2.3. The references quoted may be consulted for more details of the present topic.

2.2 Information content, Entropy

Messages produced by discrete information sources consist of sequences of symbols. For the time being, we will assume that each symbol produced by such a source has a fixed duration in time (later we will deal with sources that produce symbols of different, but fixed, length). The simplest model of an information source is probably a source that emits symbols in a statistically independent sequence, with the probabilities of occurrence of various symbols being invariant with respect to time.

2.2.1 Entropy of memoryless sources

This section applies to sources that emit symbols in statistically independent sequences. Such a source is usually called *memoryless*. We assume

that the symbol transmitted by the source is the result of a probabilistic experiment. Suppose that the probabilistic experiment involves the observation of a discrete random variable denoted by X . Let X take on one of a finite set of possible values $\{x_1, \dots, x_M\}$ with probabilities p_1, \dots, p_M , respectively. Clearly,

$$\sum_{i=1}^M p_i = 1.$$

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

We next consider a formulation of the information content in terms of information theory. It is not possible here to examine in detail the formal establishment of a measure of information. Again, the references provided at the end of this chapter may be consulted for more details of the present topic. The fundamental notion in information theory is that of surprise or uncertainty. Unlikely messages are assumed to carry more information than likely ones, and *vice versa*. Information can be measured on the basis of the amount of surprise, unpredictability or news value that it conveys to the receiver. A measure of information content of an information source, usually called *uncertainty* or *entropy*, was proposed by Shannon [296]

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

The quantity entropy measures the number of bits (per unit of time) required to send the sequences $\{X\}$ generated by the source. A translating device called *source code* is needed to "compress" the sources sequences $\{X\}$ into sequences $\{X_c\}$, whose symbols are produced at (at least) $H(\cdot)$ bits per unit of time. The quantity entropy sets a limit to the amount of compression. The base of the logarithm in (2.1) is arbitrary but will, of course, affect the numerical value of the entropy $H(p_1, \dots, p_M)$. When the logarithm is taken to the base e , the information is measured in units of *nats* (a shorthand notation of natural units). The 2-base is in common use. If we employ base 2, the units of $H(p_1, \dots, p_M)$ are expressed in bits, a convenient and easily appreciated unit of information. If $M = 2$, one usually writes $h(p)$ instead of $H(p, 1 - p)$. It follows that

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

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

Example 2.1 Consider the flipping of a coin. Let $Pr(heads) = p$ and $Pr(tails) = 1-p$, $0 \leq p \leq 1$, then the entropy is given by (2.2). That $h(1/2) = 1$ is of course intuitively confirmed by the fact that one requires 1 bit to represent the outcome of the tossing of a fair coin. For example, the outcome *heads* is represented by a logical 'zero' and *tails* by a logical 'one'. When we have an unfair coin, this simple representation method is not efficient. Consider an unfair coin with $Pr(heads) = 1/4$. Since $h(1/4) \simeq 0.811$, on the average, only 811 bits are needed to represent the outcomes of 1000 tossings with this unfair coin. How this is done in an efficient fashion is not straightforward. Codes are required to "compress" the 1000 outcomes into a message of, on the average 811 bits. Compression codes or source codes, although an important subject in its own right, are not further pursued in this book. The reader is referred to, for example, the textbooks by Blahut or Gallager [33, 108].

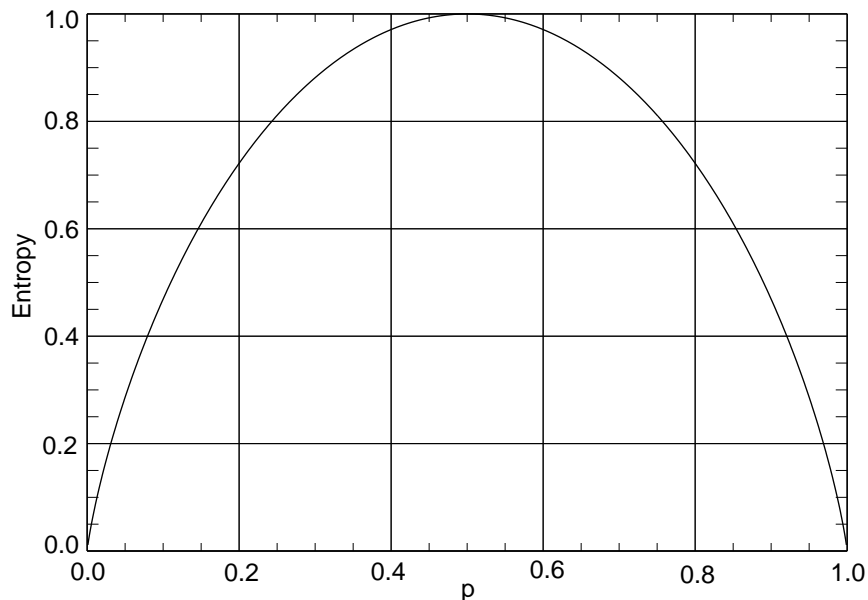


Figure 2.1: Entropy $h(p) = -p \log p - (1-p) \log(1-p)$. The information content of a stochastic variable reaches a maximum when $p = 1/2$.

We now establish two useful inequalities. Let p_1, \dots, p_M and q_1, \dots, q_M be arbitrary positive numbers with $\sum p_i = \sum q_i = 1$. Then

$$-\sum_{i=1}^M p_i \log_2 p_i \leq -\sum_{i=1}^M p_i \log_2 q_i. \quad (2.3)$$

Equality holds if and only if all $p_i = q_i$. This inequality is called the *Gibbs' inequality*. The proof is based on the observation $\log_e x \leq x - 1$, $x > 0$, with

equality if and only if $x = 1$. Thus $\log_e(q_i/p_i) \leq q_i/p_i - 1$, or $\log_2(q_i/p_i) \leq (q_i/p_i - 1) \log_2 e$. Multiplying this inequality by p_i and summing over i we obtain

$$\sum_{i=1}^M p_i \log_2 \frac{q_i}{p_i} \leq \log_2 e \sum_{i=1}^M (q_i - p_i) = 0.$$

Thus

$$-\sum_{i=1}^M p_i \log_2 p_i \leq -\sum_{i=1}^M p_i \log_2 q_i,$$

which proves the Gibbs' inequality.

It is important to know the conditions on a probability distribution that lead to maximum entropy. (Clearly, the minimum value of the entropy $H(p_1, \dots, p_M) = 0$ is taken if e.g. $p_1 = 1$ and all other $p_i = 0$.) The application of Gibbs' inequality (2.3) with all $q_i = 1/M$ yields

$$-\sum_{i=1}^M p_i \log_2 p_i \leq -\sum_{i=1}^M p_i \log_2 \frac{1}{M} = \log_2 M \sum_{i=1}^M p_i = \log_2 M$$

with equality if and only if $p_i = q_i = 1/M$ for all i . Therefore, after plugging this result into (2.1) we obtain

$$H(p_1, \dots, p_M) \leq \log_2 M. \quad (2.4)$$

Thus the maximum entropy of a memoryless source is $\log_2 M$, and this is achieved when all the p_j are equal. Any other distribution than the uniform distribution will lead to a smaller entropy of the message.

2.2.2 Markov chains

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

In probability theory the notation

$$Pr(A|B)$$

means the probability of occurrence of event A given that (|) event B has occurred. Many of the structures that will be encountered in the subsequent chapters can usefully be modelled in terms of a *Markov chain*. An N -state Markov chain is defined as a discrete random process of the form

$$\{\dots, Z_{-2}, Z_{-1}, Z_0, Z_1, \dots\},$$

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

$$Pr(Z_t = \sigma_{i_t} | Z_{t-1} = \sigma_{i_{t-1}}, Z_{t-2} = \sigma_{i_{t-2}}, \dots) = Pr(Z_t = \sigma_{i_t} | Z_{t-1} = \sigma_{i_{t-1}}).$$

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

$$q_{ij} = Pr(Z_t = \sigma_j | Z_{t-1} = \sigma_i), \quad 1 \leq i, j \leq N. \quad (2.5)$$

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

Imagine the process starts at time $t = 1$ by choosing an initial state in accordance with a specified probability distribution. If we are in state σ_i at time $t = 1$, then the process moves at $t = 2$ to a possibly new state, the quantity q_{ij} is the probability that the process will move to state σ_j at time $t = 2$. If we are in state σ_j at instant $t = 2$, we move to σ_k at instant $t = 3$ with probability q_{jk} . This procedure is repeated *ad infinitum*. A sequence of states thus generated is often called a *path*.

The state-transition diagram of a Markov chain, portrayed in Figure 2.2 represents a Markov chain as a *directed graph* or *digraph*, where the states are embodied by the *nodes* or *vertices* of the graph. The transition between states is represented by a directed line, an *edge*, from the initial to the final state. The transition probabilities q_{ij} corresponding to various transitions are shown marked along the lines of the graph. Clearly the transition matrix is

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

Another useful representation of a Markov chain is provided by a *trellis* or *lattice diagram* (see Figure 2.2). This is a state diagram augmented by a

time axis so that it provides for easy visualization of how the states change with time.

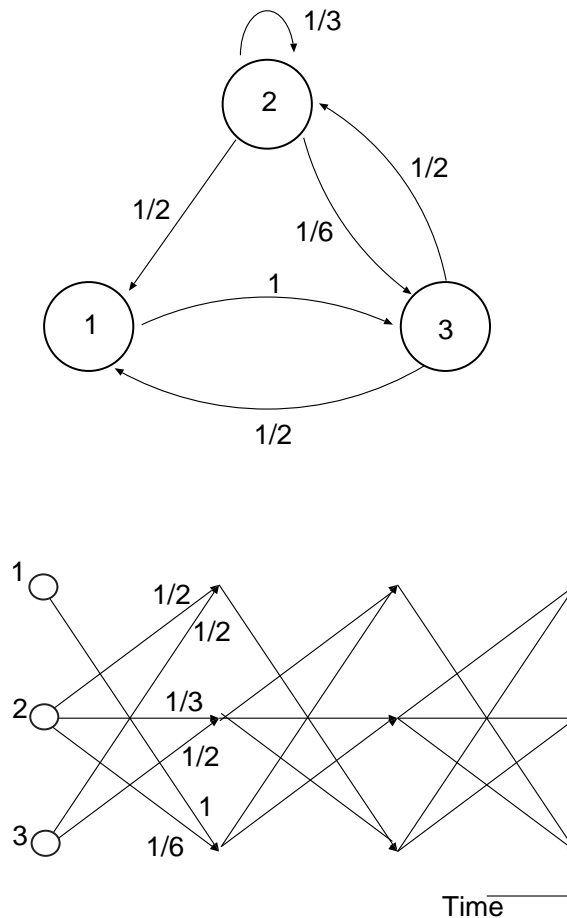


Figure 2.2: Alternative representations of a three-state Markov chain. (upper diagram) state-transition diagram, (lower diagram) trellis diagram for the same chain. Permissible transitions from one state to the other are depicted by lines.

There are many types of Markov chains, but here we restrict our attention to chains that are *ergodic* or *regular* and *irreducible* [199]. A source is irreducible if from any state the chain can eventually reach any other state (not necessarily in a single step). A state σ_i has period d if, given that we are in σ_i at instant $t = 0$, we can only reach σ_i when t is a multiple of d . We call σ_i periodic if it has some period > 1 . If the Markov chain is irreducible, either all states are periodic or none are. Regularity means that the Markov chain is non-periodic. A source is said to be ergodic if it is irreducible and regular. In the structures that will be encountered, all these conditions hold.

We shall now take a closer look at the dynamics of a Markov chain. To

that end, let $w_j^{(t)} \equiv Pr(Z_t = \sigma_j)$ represent the probability of being in state σ_j at time t . Clearly,

$$\sum_{j=1}^N w_j^{(t)} = 1. \quad (2.6)$$

The probability of being in state σ_j at time t may be expressed in the state probabilities at instant $t - 1$ as follows:

$$w_j^{(t)} = w_1^{(t-1)}q_{1j} + w_2^{(t-1)}q_{2j} + \cdots + w_N^{(t-1)}q_{Nj}. \quad (2.7)$$

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

$$\mathbf{w}^{(t)} = (w_1^{(t)}, \dots, w_N^{(t)}),$$

then the previous equation can succinctly be expressed in an elegant matrix/vector notation, thus

$$\mathbf{w}^{(t)} = \mathbf{w}^{(t-1)}Q. \quad (2.8)$$

By iteration we obtain

$$\mathbf{w}^{(t)} = \mathbf{w}^{(1)}Q^{t-1}. \quad (2.9)$$

In words, the state distribution vector at time t is the product of the state distribution vector at time $t = 1$, and the $(t - 1)$ th power of the transition matrix. It is easy to see that Q^{t-1} is also a stochastic matrix. Eq. (2.9) is equivalent to the assertion that the *n-step transition matrix* is the n th power of the single step transition matrix Q . We note also that $Q^0 = I$ is the ordinary identity matrix.

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

$$\boldsymbol{\pi} = \lim_{t \rightarrow \infty} \mathbf{w}^{(1)}Q^{t-1}. \quad (2.10)$$

The number π_i is called the *steady state probability* or *stationary state probability* of state σ_i . If the source is ergodic there is exactly one $\boldsymbol{\pi}$.

The equilibrium distribution vector can be obtained by solving the system of linear equations in the N unknowns π_1, \dots, π_N :

$$\boldsymbol{\pi}Q = \boldsymbol{\pi}. \quad (2.11)$$

Only $N - 1$ of these N equations are independent, so we solve the top $N - 1$ along with the normalizing condition

$$\sum_{i=1}^N \pi_i = 1.$$

The proof of (2.11) is elementary: we note that if $\pi Q = \pi$, then

$$\pi Q^t = \pi Q Q^{t-1} = \pi Q^{t-1} = \dots = \pi.$$

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

$$\det[Q - \lambda I] = 0, \quad (2.12)$$

where I is the identity matrix, and N (left) eigenvectors $\{\mathbf{u}_1, \dots, \mathbf{u}_N\}$ each of which is a solution of the system

$$\mathbf{u}_i Q = \lambda_i \mathbf{u}_i, \quad i = 1, \dots, N. \quad (2.13)$$

Provided that $\lambda_i, i = 1, \dots, N$, are distinct, there are N independent eigenvectors, and the eigenvectors $\mathbf{u}_i, i = 1, \dots, N$, constitute a basis. The initial state vector may be written as

$$\mathbf{w}^{(1)} = \sum_{i=1}^N a_i \mathbf{u}_i. \quad (2.14)$$

With (2.9) we find the state distribution vector $\mathbf{w}^{(t)}$ at instant t :

$$\mathbf{w}^{(t)} = \mathbf{w}^{(1)} Q^{(t-1)} = \sum_{i=1}^N a_i \lambda_i^{(t-1)} \mathbf{u}_i. \quad (2.15)$$

If the eigenvalues are distinct, the $\{\lambda_i\}$ can be ordered, such that

$$|\lambda_1| > |\lambda_2| > |\lambda_3| > \dots \text{ etc.}$$

Combination of (2.11) and (2.13) reveals that π is an eigenvector with unity eigenvalue, thus $\lambda_1 = 1$. We then have

$$\mathbf{w}^{(t)} = \pi + \sum_{i=2}^N a_i \lambda_i^{(t-1)} \mathbf{u}_i \quad (2.16)$$

and convergence to π is assured since $|\lambda_i| < 1, i \neq 1$.

2.2.3 Entropy of Markov information sources

We are now in the position to describe a *Markov information source*. Given a finite Markov chain $\{Z_t\}$ and a function ζ whose domain is the set of states of the chain and whose range is a finite set Γ , the *source alphabet*, then the sequence $\{X_t\}$, where $X_t = \zeta(Z_t)$, is said to be the output of a Markov information source corresponding to the chain $\{Z_t\}$ and the function ζ . In general, the number of states can be larger than the cardinality of the source alphabet, which means that one output symbol may correspond to more than one state. The essential feature of the Markov information source is that it provides for dependence between successive symbols, which introduces redundancy in the message sequence. Each symbol conveys less information than it is capable of conveying since it is to some extent predictable from the preceding symbol.

In the foregoing description of an information source we assumed that the symbol emitted is solely a function of the state that is entered. This type of description is usually called a *Moore-type* Markov source. In a different description, called the *Mealy-type* Markov source, the symbols emitted are a function of the Markov chain $X_t = \hat{\zeta}(Z_t, Z_{t+1})$. In other words, a Mealy-type Markov source is obtained by labeling the edges of the directed graph that represents the Markov chain. Mealy- and Moore-type descriptions are equivalent. Let a Mealy-type machine be given. By defining a Markov information source with state set composed of triples $\{\sigma_i, \sigma_j, \hat{\zeta}(\sigma_i, \sigma_j)\}$ and label $\hat{\zeta}(\sigma_i, \sigma_j)$ on the state $\{\sigma_i, \sigma_j, \hat{\zeta}(\sigma_i, \sigma_j)\}$, we obtain a Moore-type Markov source. The Moore-type source obtained by the above operation is called the *edge graph* of the Mealy-type source.

A typical example of a Mealy-type information source and its Moore-type equivalent are shown in Figure 2.3. In Figure 2.3, the symbol sequence 'addcba' can be generated by following a path in the direction of the arrows, while it can easily be seen that sequences with the subsequences 'aa' or 'cc' cannot be generated. The set of all finite sequences generated by a Markov source will be called a *constrained system* or *constraint*. We say that the Markov source represents the constrained system. The idea of a Markov source makes it possible to represent certain types of structure in streams of data.

We next examine the information content, or entropy, of a sequence emitted by a Markov source. The entropy of a Markov information source is hard to compute in most cases. For a certain class of Markov information sources, termed *unifilar* Markov information source, the computation may be greatly simplified. The word unifilar refers to the following property [7].

Let a Moore-type Markov information source with a set of states $\Sigma = \{\sigma_1, \dots, \sigma_N\}$, output alphabet Γ , and associated output function $\zeta(Z_t)$ be given. For each state $\sigma_k \in \Sigma$, let $\sigma_{k_1}, \sigma_{k_2}, \dots, \sigma_{k_{n_k}}$ be the n_k states that can

be reached in one step from σ_k , that is, the states σ_j such that $q_{kj} > 0$. We say σ_j is a *successor* of σ_k if $q_{kj} > 0$. The source is said to be unifilar if for each state σ_k the labels $\zeta(\sigma_{k_1}), \dots, \zeta(\sigma_{k_{n_k}})$ are distinct. In other words, each successor state of σ_k must be associated with a distinct symbol. Provided this condition is met and the initial state of the Markov information source is known, the sequence of emitted symbols determines the sequence of states followed by the chain, and a simple formula is available for the entropy of the emitted X -process. Similarly, a Mealy-type Markov source is said to be unifilar if for each state σ_k , $k \in \{1, \dots, M\}$ the labels tagged to the outgoing edges are distinct.

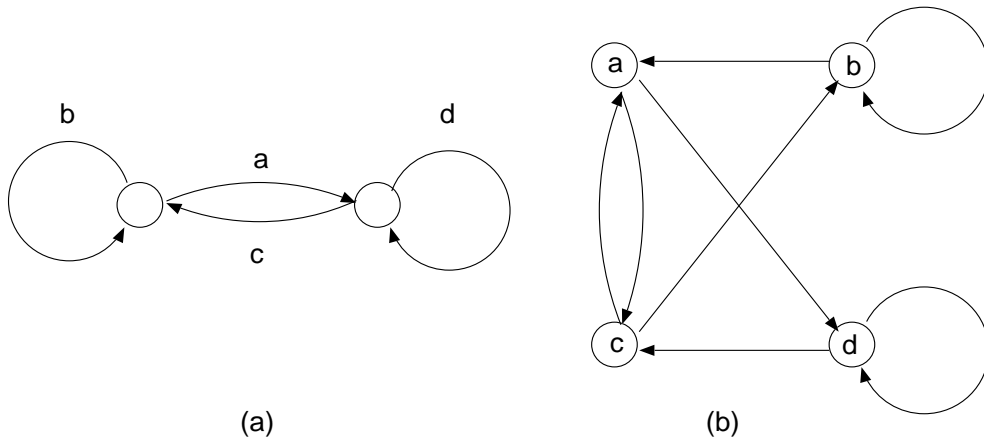


Figure 2.3: (a) Example of a Mealy-type two-state Markov information source, and (b) its four-state Moore-type counterpart.

Given a unifilar Markov source, as above, let $\sigma_{k_1}, \dots, \sigma_{k_{n_k}}$ be the successors of σ_k , then it is quite natural to define the *uncertainty* of state σ_k as $H_k = H(q_{k,k_1}, \dots, q_{k,k_{n_k}})$, with the entropy function, $H(\cdot)$, as defined in (2.1). Shannon [296] defined the entropy of the unifilar Markov source as the average of the state entropies H_k , $1 \leq k \leq N$, weighed in accordance with the steady-state probability of being in a state in question, or

$$H\{X\} = \sum_{k=1}^N \pi_k H_k. \quad (2.17)$$

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

Example 2.2 Consider the three-state Markov chain depicted in Figure 2.2, page 15. From the diagram we may read the transition probability matrix

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

What is the average probability of being in one of the three states? We find, using (2.11) the following system of linear equations that govern the steady-state probabilities:

$$\begin{aligned}\frac{1}{2}\pi_2 + \frac{1}{2}\pi_3 &= \pi_1 \\ \frac{1}{3}\pi_2 + \frac{1}{2}\pi_3 &= \pi_2 \\ \pi_1 + \frac{1}{6}\pi_2 &= \pi_3,\end{aligned}$$

from which we obtain $\pi_3 = \frac{4}{3}\pi_2$ and $\pi_1 = \frac{7}{6}\pi_2$. Since $\pi_1 + \pi_2 + \pi_3 = 1$ we have

$$\pi_1 = \frac{1}{3}, \quad \pi_2 = \frac{2}{7}, \quad \pi_3 = \frac{8}{21}.$$

The Markov chain depicted in Figure 2.2 is converted into a Markov source by assuming that it emits the letter i , $1 \leq 3$, when it visits state σ_i . It can be verified that the Markov so obtained is indeed unifilar. The entropy of the Markov information source is found with (2.1) and (2.17)

$$H\{X\} = \frac{1}{3}H(1) + \frac{2}{7}H\left(\frac{1}{2}, \frac{1}{3}, \frac{1}{6}\right) + \frac{8}{21}H\left(\frac{1}{2}, \frac{1}{2}\right) \simeq 0.798.$$

In the next section we consider a problem which is central to the field of input-constrained channels. We focus on methods to compute the maximum amount of information that can be sent over an input-constrained channel per unit of time.

2.3 Channel capacity of constrained channels

In an input-constrained, or, in short, constrained channel, given (sub)sequences are not admissible. A very simple constrained channel is for example where sequences having two consecutive 'one's are forbidden.

Shannon [296] defined the capacity C of a constrained channel by

$$C = \lim_{m \rightarrow \infty} \frac{1}{m} \log_2 N(m), \quad (2.18)$$

where $N(m)$ denotes the number of admissible signals of length m . The capacity C is a quantity that upper bounds the rate of any coding scheme that translates arbitrary data into sequences with the given constraints in force. The problem of calculation of the capacity for constrained channels is in essence a combinatorial problem, that is, of finding the number of admissible sequences $N(m)$ for large values of m . The following example will show how the counting of sequences can be done when the constraint is relatively simple.

Example 2.3 Define a channel that is allowed to transmit the two words '0' and '10'. There is only one admissible sequence of unity length, namely '0'. There are two sequences of length two, namely '00' and '10'. There are three sequences, '000', '010', and '100', of length three. Thus $N(1) = 1$, $N(2) = 2$, and $N(3) = 3$. It is not too difficult to verify that the general expression for $N(m)$, $m > 2$, is given by the recurrence expression

$$N(m) = N(m-1) + N(m-2).$$

We can write down the following explicit expression for $N(m)$ (see Chapter 4):

$$N(m) = \frac{1}{\sqrt{5}} \left(\frac{1 + \sqrt{5}}{2} \right)^{m+1} - \frac{1}{\sqrt{5}} \left(\frac{1 - \sqrt{5}}{2} \right)^{m+1}, \quad m \geq 0.$$

For large values of the sequence length m we can approximate $N(m)$ by

$$N(m) \approx \frac{1}{\sqrt{5}} \left(\frac{1 + \sqrt{5}}{2} \right)^{m+1},$$

so that after invoking definition (2.18) we find the capacity C

$$C = \lim_{m \rightarrow \infty} \frac{\log_2 N(m)}{m} = \log_2 \frac{1 + \sqrt{5}}{2} \simeq 0.694.$$

When the constraints are more complicated than in the above trivial example, we have to resort to a larger arsenal of mathematical tools. We start, since virtually all channel constraints can be modelled as such, with the computation of the capacity of Markov information sources.

2.3.1 Capacity of Markov information sources

In the previous sections we developed a measure of information content of an information source that can be represented by a finite Markov model. As discussed, the measure of information content, entropy, can be expressed in terms of the limiting state-transition probabilities and the conditional entropy of the states. In this section we address a problem that provides the key to answer many questions that will emerge in the chapters to follow. Given a unifilar N -state Markov source with states $\{\sigma_1, \dots, \sigma_N\}$ whose transition probability matrix is irrelevant, we define the *connection matrix* $D = \{d_{ij}\}$ of the source as follows. For an N -state source, the connection (or adjacency) matrix D is defined by $d_{ij} = n_{ij}$ where n_{ij} is the number of edges going from state i to state j . In other word, D is the connection matrix of the directed graph underlying the Markov source.

For a given connection matrix, we wish to choose the transition probabilities in such a way that the entropy

$$H\{X\} = \sum_{k=1}^N p_k H_k$$

is maximized. A Markov source that has the above transition probabilities is called *maxentropic*, and sequences generated by such a maxentropic unifilar source are called *maxentropic sequences*. The study of the statistical properties, such as spectra, provides useful tools for assessing the performance of constrained codes.

The maximum entropy of a unifilar Markov information source, given its connection matrix, is given by [296]

$$C = \max H\{X\} = \log_2 \lambda_{\max}, \quad (2.19)$$

where λ_{\max} is the largest eigenvalue of the connection matrix D . The existence of a positive eigenvalue and corresponding eigenvector with positive elements is guaranteed by the Perron-Frobenius theorems [325]. Essentially, there are two approaches to prove the preceding equation. One approach, provided by Shannon [296] is a straightforward routine, using Lagrange multipliers, of finding the extreme value of a function of several independent variables. The second proof of (2.19) to be followed here, is established by enumerating the number of distinct sequences that a Markov source can generate.

As it is assumed that the Markov source is unifilar, we need to compute the number of paths that lead from one state to another (or the same) state. Matrix operations help us to count the number of paths. The following theorem is very useful.

Theorem 2.1 *If D is the $N \times N$ one-step connection matrix of a given digraph, then the (i, j) -th entry of D^m equals the number of paths of length m that lead from state σ_i to state σ_j , $1 \leq i, j \leq N$.*

Proof: Let $[D]_{ij}^m$ denote the (i, j) -th entry of D^m . The number of paths of length m that lead from state σ_i to state σ_j equals the number of paths of length $m - 1$ from state σ_i to state σ_h times the number of path of unity length from σ_h to σ_j . This number is

$$[D]_{ih}^{m-1}[D]_{hj}.$$

Then, by induction, the number of paths of length m from σ_i to σ_j equals

$$\sum_{h=1}^N [D]_{ih}^{m-1}[D]_{hj} = [D]_{ij}^m,$$

which concludes the proof.

For large sequence length m we may conveniently approximate the number of sequences, $[D]_{ij}^m$, by

$$[D]_{ij}^m \simeq a_{ij} \lambda_{\max}^m, \quad (2.20)$$

where a_{ij} is a constant, and λ_{\max} is the largest real eigenvalue of the matrix D [109]. In other words, λ_{\max} is the largest real root of the determinant equation

$$\det[D - zI] = 0. \quad (2.21)$$

Equation (2.20) states that, for large enough m , the number of distinct sequences grows exponentially with the sequence length m . The growth factor is λ_{\max} . This is not to say that $[D]_{ij}^m$ is accurately determined by the exponential term when m is small. We have

$$\frac{1}{m} \log_2 [D]_{ij}^m \simeq \frac{1}{m} (\log_2 a_{ij} + m \log_2 \lambda_{\max}).$$

The maximum entropy of the noiseless channel may be evaluated by invoking (2.18) or

$$C = \lim_{m \rightarrow \infty} \frac{1}{m} \log_2 [D]_{ij}^m = \log_2 \lambda_{\max},$$

which concludes the proof of (2.19).

Computation of the maxentropic transition probabilities

The computation of the transition probabilities q_{ij} associated with the maximum entropy of the source is quite relevant for our future purposes. It allows us to compute, for example, the spectral properties of maxentropic sequences using the theory developed in the next chapter. The transition probabilities can be found with the following reasoning. Let $\mathbf{p} = (p_1, \dots, p_N)^T$ denote the eigenvector associated with the eigenvalue λ_{\max} , or

$$D\mathbf{p} = \lambda_{\max}\mathbf{p}. \quad (2.22)$$

The state-transition probabilities that maximize the entropy are

$$q_{ij} = \lambda_{\max}^{-1} d_{ij} \frac{p_j}{p_i}. \quad (2.23)$$

To prove (2.23) is a matter of substitution. According to the Perron-Frobenius theorems [325] the components of the eigenvector \mathbf{p} are positive, and thus $q_{ij} \geq 0$, $1 \leq i, j \leq N$. Since $\mathbf{p} = (p_1, \dots, p_N)^T$ is an eigenvector for λ_{\max} , we conclude

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

and hence the matrix Q is indeed stochastic.

In the following paragraph we will demonstrate that the transition probabilities given by (2.23) are indeed maximizing the entropy. The entropy of a Markov information source is, according to definition (2.17)

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

where H_k is the uncertainty of state σ_k and (π_1, \dots, π_N) is the steady-state distribution. Thus,

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

Since

$$\sum_{i,j} \pi_i q_{ij} \log_2 p_i = \sum_i \pi_i \log_2 p_i$$

and

$$\sum_{i,j} \pi_i q_{ij} \log_2 p_j = \sum_j \log_2 p_j \sum_i \pi_i q_{ij} = \sum_j \pi_j \log_2 p_j,$$

we obtain

$$H\{X\} = - \sum_{i,j=1}^N \pi_i q_{ij} \log_2 q_{ij} = \sum_{i,j} \pi_i q_{ij} \log_2 \lambda_{\max} = \log_2 \lambda_{\max}.$$

This demonstrates that the transition probabilities given by (2.23) are indeed maximizing the entropy.

Example 2.4 We revert to the three-state unifilar Markov chain discussed in Example 2.2. The adjacency matrix, D , is

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

The characteristic equation is

$$\begin{aligned} \det[D - zI] &= -z(z^2 - z - 2) \\ &= -z(z - 2)(z + 1) = 0, \end{aligned}$$

from which we conclude that the largest root is $\lambda_{\max} = 2$, and the capacity is $C = \log_2 \lambda_{\max} = 1$. The eigenvector associated with the largest eigenvalue is $\mathbf{p} = (1, 3, 2)^T$. The transition probabilities that maximize the entropy of the Markov information source are found with (2.23)

$$Q = \begin{bmatrix} 0 & 0 & 1 \\ \frac{1}{6} & \frac{1}{2} & \frac{1}{3} \\ \frac{1}{4} & \frac{3}{4} & 0 \end{bmatrix}.$$

2.3.2 Sources with variable-length symbols

A basic assumption made in the previous sections is that all transmitted symbols are of unity duration. Our endeavor in this section is to extend the results on channels whose symbols are of unity duration to channels whose symbols do not all have the same duration. In particular, we shall consider

sequences that are composed of elementary symbols, called *phrases*, selected from a finite set $\{x_1, \dots, x_M\}$. Each of the symbols x_1, \dots, x_M is assumed to have a certain duration (or length) t_1, \dots, t_M that are integer multiples of the unit of time.

If the source is memoryless, and the probability that phrase x_i is sent equals p_i then the entropy $H(p_1, \dots, p_M)$ is given by [296]

$$H(p_1, \dots, p_M) = -\frac{\sum_{i=1}^M p_i \log_2 p_i}{\sum_{i=1}^M t_i p_i}, \quad 0 \leq p_i \leq 1. \quad (2.24)$$

Similar relations can be written down for sources with memory. The capacity computation of a source that can be modelled with variable length symbols was also addressed by Shannon. Below we will extend the capacity computations of Markov sources generating variable length phrases.

We examine again an N -state Markov information source. From any state, phrases can emerge of certain, given, lengths. Let S_{ij} denote the set of the lengths of the phrases that emanate from state i and are allowed to terminate in state j . Define the $N \times N$ connection matrix $D(z)$ by

$$d_{ij}(z) = \sum_{t \in S_{ij}} z^{-t}, \quad 1 \leq i, j \leq N. \quad (2.25)$$

By methods similar to those used in deriving (2.21) it was shown by Shannon that the capacity of the above Markov source equals the base-2 logarithm of the largest real root of the characteristic equation

$$\det[D(z) - I] = 0. \quad (2.26)$$

Equation (2.26) reduces to (2.21) if all symbols are of unity duration.

If the phrases are emitted independently, then (2.25) and (2.26) reduce to the characteristic equation

$$P(z) = z^{-t_1} + \dots + z^{-t_M} = 1. \quad (2.27)$$

Alternatively, the above result can be obtained by maximizing (2.24) using Lagrange multipliers under the condition that $\sum p_i = 1$. From (2.27) it is clear that $P(z)$ is a monotonic decreasing function with

$$P(0) = \infty, \quad P(\infty) = 0.$$

Therefore, the equation $P(z) = 1$ has exactly one positive root. It is instructive at this point to study a simple case.

Example 2.5 Consider first a sequence composed of symbols of duration two, three, and four time units, respectively, which can be transmitted independently.

This simple source is displayed in Figure 2.4. Then, according to (2.27), the characteristic equation is

$$z^{-2} + z^{-3} + z^{-4} = 1,$$

or

$$z^4 - z^2 - z - 1 = 0.$$

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

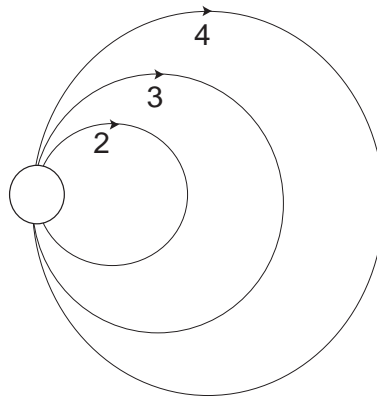


Figure 2.4: Simple source that transmits symbols of duration two, three, and four time units.

In the next example we assume the same set of phrases, but now we apply the extra proviso that the phrases are not emitted independently.

Example 2.6 As in the previous example, sequences are composed of phrases of length two, three, and four time units. A phrase of length two is not allowed to follow a phrase of length two. Figure 2.5 shows a Moore-type diagram of this source.

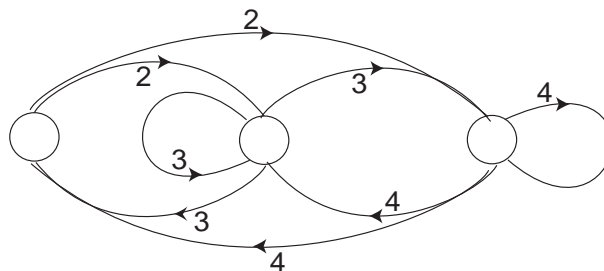


Figure 2.5: Simple source that transmits symbols of duration two, three, and four time units, but a symbol of length two may not follow a symbol of length two.

We can write down the following connection matrix

$$D(z) = \begin{bmatrix} 0 & z^{-2} & z^{-2} \\ z^{-3} & z^{-3} & z^{-3} \\ z^{-4} & z^{-4} & z^{-4} \end{bmatrix}.$$

Invoking (2.26) we obtain the equation

$$\det \begin{bmatrix} -1 & z^{-2} & z^{-2} \\ z^{-3} & z^{-3} - 1 & z^{-3} \\ z^{-4} & z^{-4} & z^{-4} - 1 \end{bmatrix} = 0,$$

or

$$z^6 - z^3 - z^2 - z - 1 = 0.$$

The capacity of this channel is 0.465.

The following example offers an overview of the approaches for computing the capacity of a constrained channel.

Example 2.7 Assume we have sequences that have an even number of 'zero's between consecutive 'one's. What is the capacity of this constrained channel? Below we give three worked approaches to solve this problem.

a) *Combinatorics*: After a little thought we may see that the number of allowed sequences $N(n)$ is

$$N(n) = N(n-1) + N(n-2) + 1,$$

with $N(0) = 1$ and $N(1) = 2$. The characteristic equation is therefore

$$1 = z^{-1} + z^{-2}$$

or

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

The largest real root is $\frac{1+\sqrt{5}}{2}$, so that the capacity C is

$$C = \log_2 \frac{1 + \sqrt{5}}{2} \approx 0.694.$$

The reader may easily get the impression that 0.694 is a kind of physical constant in Information Theory as it is also the capacity of the source described in Example 2.3.

b) *Markov source approach*: The constraints can be cast into the 2-state labelled directed graph portrayed in Figure 2.6. Any path through the graph stepping from state to state, and reading off the labels attached to the edges connecting the states, yields an allowed sequence.

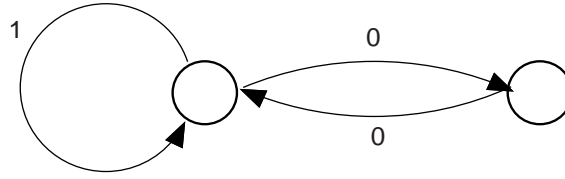


Figure 2.6: Two-state Mealy-type source that generates sequences with an even number of '0's between consecutive '1's.

The transition matrix is

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

The characteristic equation is

$$\det[D - zI] = z^2 - z - 1 = 0.$$

c) *Runlength approach*: An allowed sequence can be thought to be built up from phrases '1', '100', '10000', and so on. The characteristic equation is therefore

$$z^{-1} + z^{-3} + z^{-5} + \dots = 1$$

or

$$z^{-1} + z^{-3} \left(\frac{1}{1 - z^{-2}} \right) = 1$$

or

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

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

Chapter 3

Spectral Analysis

3.1 Introduction

The purpose of this chapter is to provide a summary of the art and a tutorial exposition of the spectral analysis of signals whose statistics are stationary. After the preliminary notions, a spectral analysis of block-coded sequences is presented. The spectral analysis of block-coded sequences is compounded by the fact that its statistical properties are, in general, cyclo-stationary rather than stationary, that is, the statistics will vary periodically with a period equal to the duration of a block.

Any time-varying signal can be represented by a set of frequency components which occupy a certain range of frequencies. The frequency components form the *spectrum* of the signal and are given by the Fourier transform of the signal waveform. A significant statistic of a stochastic signal is its *power spectral density function*. It is, therefore, both convenient and appropriate to begin with a chapter that briefly considers the spectral properties of signals constituted by coded sequences. It is central to the coding for input-constrained channels, and plays an important part in the chapters to follow. Specifically, we shall concentrate on the evaluation of the spectrum of sequences emitted by a Markov information source. The last sections of this chapter deal with the spectral properties of sequences of block-coded signals. The spectral analysis of block codes presented here is derived following the method of Cariolaro, Pierobon, and Tronca [48, 49].

Spectral analysis of random signals is a topic amply developed elsewhere (see e.g. the textbook by Papoulis [271]) and the outline is, therefore, very brief and in most cases results are given without proof. The following sections serve mainly to introduce a glossary of concepts and notation that will be employed later. It is assumed that the reader has a basic background in probability theory and random variables. We shall avoid as much of the detailed mathematical analysis as possible and concentrate upon those areas where the theory has produced results of relevance to spectral shaping

codes. The reader who wishes a more leisurely introduction is encouraged to consult the references quoted for more details of the present topic.

3.2 Stochastic processes

Let a stochastic process X_t be defined as an ensemble of sample functions, then we may consider the values of the process at any set of time instants $t_1 < t_2 < \dots < t_n$. The random variables X_{t_i} , $i = 1, \dots, n$, are characterized statistically by their joint probability density function (pdf)

$$p(x_{t_1}, \dots, x_{t_n}).$$

For a specific value of t_i the random variable $X(t_i)$ can be characterized by a first-order probability density function, which is denoted by $p(x_{t_i})$. The first-order moments of $X(t_i)$ are defined by

$$M_X^{(n)}(t_i) = E\{X(t_i)^n\} = \int_{-\infty}^{\infty} x_{t_i}^n p(x_{t_i}) dx_{t_i}, \quad (3.1)$$

where $E\{\cdot\}$ designates *expectation* or *ensemble average*. In general, the n th moment will depend on the instant t_i if the pdf of X_{t_i} depends on t_i . Of particular interest is the mean value of the process, which is given by $M_X(t_i) = E\{X(t_i)\}$.

Next we consider the two random variables X_{t_1} and X_{t_2} . The correlation between the two random variables is measured by the joint moments

$$E\{X_{t_1}^n X_{t_2}^m\} = \int_{-\infty}^{\infty} \int_{-\infty}^{\infty} x_{t_1}^n x_{t_2}^m p(x_{t_1}, x_{t_2}) dx_{t_1} dx_{t_2}. \quad (3.2)$$

For second-order statistics we are particularly interested in the *auto-correlation function*, which is defined by

$$R_X(t_1, t_2) = E\{X_{t_1} X_{t_2}\}. \quad (3.3)$$

The value of $R_X(t_1, t_2)$ for $t_1 = t_2 = t$ is the *average power* of the process X_t :

$$E\{X_t^2\} = R_X(t, t). \quad (3.4)$$

The auto-covariance function is defined by

$$C_X(t_1, t_2) = E\{(X_{t_1} - M_X(t_1))(X_{t_2} - M_X(t_2))\}. \quad (3.5)$$

3.2.1 Stationary processes

A process is said to be *stationary in the strict sense* if all its statistics are time invariant. When the stochastic process X_t is stationary, the pdf of

the pair (X_{t_1}, X_{t_2}) is identical to the pdf of (X_{t_1+t}, X_{t_2+t}) . Therefore the auto-correlation function of a stationary process does not depend on the specific instants t_1 and t_2 , but it depends *only* on the magnitude of the time interval between t_1 and t_2 , that is, $R_X(t_1, t_2) = R_X(t_2, t_1)$. Thus,

$$R_X(t_1, t_2) = E\{X_{t_1}X_{t_2}\} = R_X(|t_1 - t_2|) = R_X(|\tau|), \quad (3.6)$$

where $\tau = t_1 - t_2$. For symmetry reasons

$$R_X(\tau) = R_X(-\tau). \quad (3.7)$$

A process is said to be *wide-sense stationary* if only mean and auto-correlation function are time invariant, that is, if

$$\begin{aligned} E\{X_t\} &= M_X, \\ E\{X_tX_{t+\tau}\} &= R_X(\tau). \end{aligned} \quad (3.8)$$

Consequently, wide-sense stationarity is a less stringent condition than strict-sense stationarity. When the process is stationary, the auto-covariance function simplifies into

$$C_X(\tau) = R_X(\tau) - M_X^2. \quad (3.9)$$

The power spectral density function is an important characteristic providing information concerning power content at low and high frequencies, average power, and bandwidth. It provides a means of determining the average interference suffered by embedded tracking, timing, and focus servo mechanisms due to the data signal and crosstalk from the signal on adjacent tracks. The Fourier transform of the auto-correlation function, called the (two-sided) power *spectrum* or *power spectral density function* of the process, is given by

$$H_X(\omega) = \int_{-\infty}^{\infty} R_X(\tau)e^{-j\omega\tau} d\tau, \quad (3.10)$$

where $j = \sqrt{-1}$. Since $R_X(\tau) = R_X(-\tau)$, and $R_X(\tau)$ is real, we conclude that $H_X(\omega)$ is real and an even function of ω . It can also be shown, see for example Papoulis [271], that the spectrum $H_X(\omega)$ is non-negative for all ω , which, by the way, is clearly a necessary condition for the interpretation of $H_X(\omega)$ as power density to be meaningful. The auto-correlation function can be found from the inverse Fourier transform relationship

$$R_X(\tau) = \frac{1}{2\pi} \int_{-\infty}^{\infty} H_X(\omega)e^{j\omega\tau} d\omega. \quad (3.11)$$

The previous equations are known as the *Wiener-Kintchine relations*. More insight can be gained by defining the power spectrum $H_X(\omega)$ as a limit of

$$H_X(\omega) = \lim_{T \rightarrow \infty} \frac{1}{2T} \left| \int_{-T}^T x(t)e^{-j\omega t} dt \right|^2, \quad (3.12)$$

where $x(t)$ is a specific realization of the process. Obviously, $H_X(\omega)$ is non-negative.

3.2.2 Cyclo-stationary processes

Many stochastic processes that we will encounter in the next chapters are processes whose statistics are periodic rather than time invariant. A process whose statistics such as symbol probability, correlation between symbols, etc., vary periodically with some period is said to be *cyclo-stationary*. This phenomenon may arise, for instance, in synchronous pulse amplitude modulated systems where information is transmitted in fixed-length time slots. For a comprehensive discussion of cyclo-stationary processes the reader is referred to the book by Franks [98], the paper by Bennett [26] or the paper by Bosik [40]. A wide-sense cyclo-stationary process satisfies the following relationship:

$$\begin{aligned} M_X &= E\{X(t)\} = M_X(t + T) \\ R_X(t, t + \tau) &= R_X(t + T, t + T + \tau), \end{aligned} \quad (3.13)$$

where T , the smallest time interval giving equality, is called the *period*. As an illustration, consider a synchronous pulse amplitude modulated signal. The modulation is said to be *synchronous* because of the uniform spacing between successive pulses. For a sound understanding of pulse amplitude modulation the reader is referred to the book by Cattermole [52].

Suppose that a message sequence $\{x_j : j \in \mathbf{Z}\}$ is transmitted with a standard pulse shape $s(t)$ for each digit and a standard period T_b . The general expression for a modulated signal is then

$$X(t) = \sum_{j=-\infty}^{\infty} x_j s(t - jT_b), \quad (3.14)$$

where $\{x_j\}$ is assumed to be a wide-sense stationary sequence, such that

$$\begin{aligned} E\{x_j\} &= \bar{x} \\ E\{x_j x_{j+k}\} &= R_x(k). \end{aligned} \quad (3.15)$$

It should be appreciated that $R_x(k)$ is a time-discrete auto-correlation function. Working out yields

$$E\{X(t)\} = \sum_j E\{x_j\} s(t - jT_b) = \bar{x} \sum_j s(t - jT_b) \quad (3.16)$$

and

$$\begin{aligned} R_X(t, t + \tau) &= \sum_i \sum_j E\{x_i x_j\} s(t - iT_b) s(t + \tau - jT_b) \\ &= \sum_k R_x(k) \sum_i s(t - iT_b) s(t + \tau - (i+k)T_b). \end{aligned} \quad (3.17)$$

The sum over i is periodic with period T_b , and it can be seen that

$$\begin{aligned} E\{X(t)\} &= E\{X(t + T_b)\} \text{ and} \\ R_X(t, t + \tau) &= R_X(t + T_b, t + T_b + \tau), \end{aligned}$$

from which we conclude that the signal $X(t)$ is wide-sense cyclo-stationary with period T_b .

A useful technique for our purposes is to define an equivalent stationary process whose statistics are those of the cyclo-stationary process averaged over the period T . The cyclo-stationary process $X(t)$ is changed into a stationary process $X_\Delta(t)$ by adding a random variable Δ , uniformly distributed over $0 \leq \Delta < T$, such that

$$X_\Delta(t) = X(t - \Delta), \quad (3.18)$$

and defining the spectral density of the cyclo-stationary process as the Fourier transform of the auto-correlation of the stationary process $X_\Delta(t)$. The physical justification of this approach (see Cattermole and O'Reilly [54]) is that, if measurements are made without a time reference, these are the statistics which will be normally observed. A practical spectrum analyzer comprises a bank of narrow-band filters whose output is averaged over a period much longer than the period times of the signals. Thus, if we assume that Δ is uniformly distributed over $0 \leq \Delta < T_b$, then we have

$$\begin{aligned} E\{X_\Delta(t)\} &= \bar{x} \sum_i \frac{1}{T_b} \int_0^{T_b} s(t - iT_b - \xi) d\xi \\ &= \bar{x} \frac{1}{T_b} \int_{-\infty}^{\infty} s(t) dt \text{ is constant} \end{aligned} \quad (3.19)$$

and

$$\begin{aligned} \tilde{R}_X(\tau) &= E\{X_\Delta(t)X_\Delta(t + \tau)\} \\ &= \sum_k R_x(k) \sum_i \frac{1}{T_b} \int_0^{T_b} s(t - iT_b - \xi)s(t + \tau - (i + k)T_b - \xi) d\xi \\ &= \sum_k R_x(k)r_s(\tau + kT_b), \end{aligned} \quad (3.20)$$

where

$$r_s(\tau) = \frac{1}{T_b} \int_{t=-\infty}^{\infty} s(t + \tau)s(t) dt. \quad (3.21)$$

The same result could have been obtained by simply averaging the auto-correlation function over one period (in t).

For the stationary case there is also a simple relationship between the power spectral density functions of the process $\{x_i\}$ and $s(t)$. To show this, we take the Fourier transform of $\tilde{R}_X(\tau)$, given by (3.20), and obtain

$$H_{X_\Delta}(\omega) = H_x(\omega)H_s(\omega), \quad (3.22)$$

where

$$H_x(\omega) = \sum_{k=-\infty}^{\infty} R_x(k)e^{-jk\omega T_b}$$

and

$$H_s(\omega) = \int_{-\infty}^{\infty} r_s(t) e^{-j\omega t} dt.$$

Thus, the power spectral density function of the phase-averaged process is obtained by multiplying the power spectral density function of the process $\{x_i\}$ by the *shaping function*, or energy spectrum, $H_s(\omega)$. From the previous equation, it is clear that the overall spectral characteristic of the process $\{X\}$ can be tailored by designing an appropriate $s(t)$ and the correlation characteristics of the sequence $\{x_i\}$. In various communication channels, such as conventional optical or magnetic recorders, we do not have the freedom of conditioning the standard pulse shape $s(t)$. In this case, the spectral density function of the process $\{X\}$ can only be shaped to certain requirements by providing dependence between consecutive symbols of the channel sequence $\{x_i\}$. This is the province of the spectral shaping codes.

The write signal $s(t)$ in conventional recorders is a full- T_b pulse with a pulse shape given by

$$p_{T_b}(t) = \begin{cases} 1, & 0 \leq t < T_b, \\ 0, & \text{otherwise.} \end{cases} \quad (3.23)$$

In this particular case, we find

$$H_s(\omega) = \int_{-\infty}^{\infty} r_s(t) e^{-j\omega t} dt = \left(\frac{\sin \omega T_b/2}{\omega T_b/2} \right)^2 T_b. \quad (3.24)$$

In the sequel we assume, for clerical convenience, unity symbol time interval, or $T_b = 1$.

The subsequent sections provide an evaluation of the power spectral density function of sequences emitted by Markov information sources and of block-coded sequences.

3.3 Spectra of Markov sources

Suppose we are given the output of an N -state Markov information source $\{X_t : t \in \mathbf{Z}\}$. Given the transition matrix, Q , of the Markov source, one can always select the initial state distribution vector in such a way that the resulting chain is stationary. Invoking the definitions given in Chapter 2, and by virtue of the assumed ergodicity of the Markov chain described therein, we can write down the auto-correlation (see Bilardi, Padovani & Pierobon [31] and Galko & Pasupathy [107])

$$R_x(k) = E\{X_t X_{t+k}\} \quad (3.25)$$

of the emitted sequence. Let $w_i^{(t)}$ be the probability that the chain is in state σ_i at the symbol interval t , then we can represent the probability that

the system is in state j at the $(t+k)$ th, $k \geq 0$, interval by

$$w_j^{(t+k)} = \sum_{i=1}^N [Q]_{ij}^k w_i^{(t)},$$

where $[Q]_{ij}^k$ denote the entries of Q^k . In the long run, the state distribution vector converges to the equilibrium distribution vector from any valid initial state probability vector $\mathbf{w}^{(1)}$, thus

$$\boldsymbol{\pi} = \lim_{t \rightarrow \infty} \mathbf{w}^{(1)} Q^{t-1}. \quad (3.26)$$

The equilibrium distribution vector is obtained by solving the system of linear equations in the N unknowns π_1, \dots, π_N :

$$\boldsymbol{\pi} Q = \boldsymbol{\pi}, \quad (3.27)$$

along with the normalizing condition

$$\sum_{i=1}^N \pi_i = 1.$$

Assuming a Moore-type Markov source (see Chapter 2), the symbol emitted when the chain visits state σ_i is $\zeta(\sigma_i)$, $i = 1, \dots, N$. Taking account of the steady-state probabilities and averaging yields

$$M_x = E\{X_t\} = \sum_{i=1}^N \zeta(\sigma_i) \pi_i. \quad (3.28)$$

From the Markov condition we conclude

$$\begin{aligned} C_x(k) &= E\{X_t X_{t+k}\} - M_x^2 \\ &= \sum_{i=1}^N \sum_{j=1}^N \pi_i \zeta(\sigma_i) \zeta(\sigma_j) [Q]_{ij}^{|k|} - \left(\sum_{i=1}^N \zeta(\sigma_i) \pi_i \right)^2 \\ &= \sum_{i=1}^N \sum_{j=1}^N \pi_i \zeta(\sigma_i) ([Q]_{ij}^{|k|} - \pi_j) \zeta(\sigma_j). \end{aligned} \quad (3.29)$$

The preceding equations can be written more conveniently in matrix notation. Let $\boldsymbol{\zeta}$ be the column vector

$$\boldsymbol{\zeta}^T = (\zeta(\sigma_1), \dots, \zeta(\sigma_N))$$

and Π the diagonal matrix $\Pi = \text{diag}\{\pi_1, \dots, \pi_N\}$. Let further

$$Q_\infty = \mathbf{1}\boldsymbol{\pi}, \quad (3.30)$$

where $\mathbf{1}$ is a column vector of all ones. Thus Q_∞ has all its rows equal to the steady-state probability vector $\boldsymbol{\pi}$. Now the auto-covariance function of the emitted sequence can be written as

$$C_x(k) = \boldsymbol{\zeta}^T \Pi(Q^{|k|} - Q_\infty) \boldsymbol{\zeta}. \quad (3.31)$$

The power spectral density function of the X -process is

$$\begin{aligned} H_x(\omega) &= \sum_{k=-\infty}^{\infty} R_x(k) e^{-jk\omega} \\ &= M_x^2 2\pi \delta(\omega) + C_x(0) + 2 \sum_{k=1}^{\infty} C_x(k) \cos k\omega. \end{aligned} \quad (3.32)$$

Before we will proceed with the spectral analysis of sequences generated by channel encoders, we will give a detailed description of the operating principles of encoders. As encoders are closely related to their inverting counterpart, the decoders, we will discuss them as well.

3.4 Description of encoder models

Physically, an encoder can be interpreted as a device with an m -bit input, an n -bit output, and an internal state. The principle of operation of an encoder can be represented by a conventional finite-state sequential machine (FSSM), a well-known concept in the fields of computation and automation theory. The specific codeword transmitted by the encoder is, in general, a function of the m -bit source word that enters the encoder and depends further on the particular state of the encoder. The sequential machine is defined in terms of three sets: the inputs, the outputs and the states, and two logical functions: the *output function* and the *next-state function*. The principles of operation of an encoder, taken from Cattermole and O'Reilly [54], can be described as follows:

1. The input set B of m -digit (binary) words.
We assume the source code to be non-redundant, so that $|B| = 2^m \equiv M$. A source word will be denoted by the symbol $\boldsymbol{\beta}_u$, $u = 0, 1, \dots, M - 1$. The k th source word in an input sequence will be denoted by \mathbf{b}_k (k any integer).
2. The set Σ of states.
We assume this to be finite, with $|\Sigma| \equiv N$. A state considered in the abstract will be denoted by σ_i , ($i = 1, 2, \dots, N$). The state at the time the k th source word is translated will be denoted by s_k .

3. The output set X of n -digit codewords.

It is convenient to identify output codewords by a double index: the word χ_{iu} is the translation of the source word β_u which is delivered when the encoder is in state σ_i . Consequently, $|X| \leq NM$. Not all these words have to be distinct, since the same words may be used in more than one state. The foregoing notation applies to codewords in the abstract: the k th codeword in an output sequence will be denoted by \mathbf{x}_k .

4. The output function h has domain $\Sigma \times B$ and range X . It specifies the translation $\chi_{iu} = h(\sigma_i, \beta_u)$. In operation,

$$\mathbf{x}_k = h(s_k, \mathbf{b}_k). \quad (3.33)$$

The output function is commonly chosen such that it has a special form of inverse $h^{-1}(\chi_{iu}) = \beta_u$. That is, knowledge of χ must identify β without reference to the state σ , to ensure unique decodability.

5. The next-state function g has domain $\Sigma \times B$ and range Σ . It specifies the state in which the encoder is left after translating one source word, so that

$$s_{k+1} = g(s_k, \mathbf{b}_k). \quad (3.34)$$

This definition accords with the usual finite-state machine convention, but in our context a useful state-transition function may be defined with the output, rather than the input, as an argument. This function f has domain $X \times \Sigma$ and range Σ , and defines the next state as

$$s_{k+1} = f(s_k, \mathbf{x}_k). \quad (3.35)$$

Given the attribute of unique decodability, these approaches are equivalent, but since the states of an encoder derive directly from the history of the output, the second version may be simpler.

In a *Moore-type finite-state machine* the codeword \mathbf{x}_k is a function of the state only (and is independent of the source word), or

$$\mathbf{x}_k = h(s_k). \quad (3.36)$$

Mealy machines can be dealt with as Moore machines, provided that we consider as a new state the pair (σ_i, β_u) . This structure offers some simplification in the characterizing function representation. Figure 3.1 portrays a block diagram of a Mealy-type finite-state sequential machine.

The description of the encoder is reminiscent of the Markov information source described in Chapter 2. It has to be borne in mind that a Markov information source is an autonomous machine without 'inputs' whereas an

encoder is a function that translates input words into an encoded output sequence. The sequence of encoder states, if the inputs are independent, is a stationary Markov chain, since the 'next' state depends only on the previous state and on the input which is, by assumption, independent of previous states or inputs.

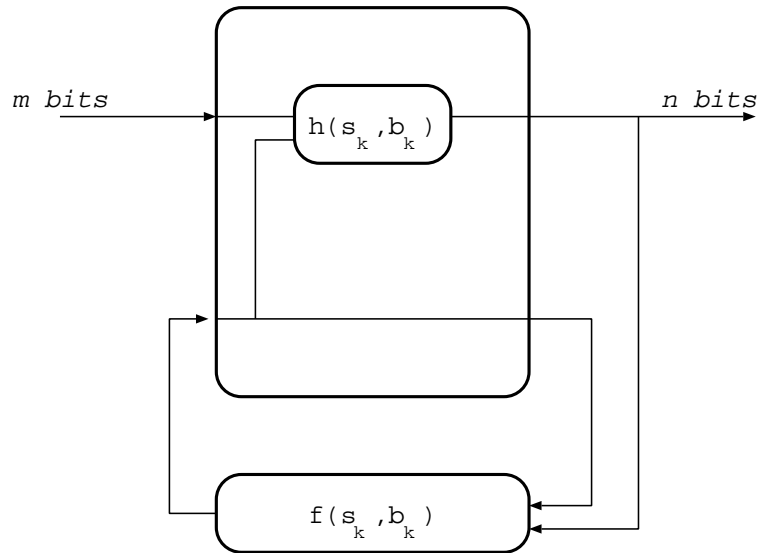


Figure 3.1: Block diagram of Mealy-type finite-state sequential machine.

The functions $g()$ and $h()$ defining the FSSM are called the *characterizing functions*. They can be specified by means of a table, called a *transition table*, or in our context called a *codebook*. For $1 \leq i \leq N$, the function $h_i : B \rightarrow X$ defined by $h_i(\beta_u) = h(\sigma_i, \beta_u)$ is called the i -th code page. Alternatively, these functions can be specified by means of an oriented graph, called *state-transition diagram*. The advantage of the state-transition diagram is both conceptual and computational. It enables one to visualize an FSSM as a mechanism which traces paths along a graph in accordance with the input sequence. Given an initial state and an input sequence, it also enables one to determine the corresponding state and output sequences by inspection of the labels.

Example 3.1 Table 3.1 shows the complete specification of a code that can be modelled with a 3-state Mealy-type encoder. The encoder translates 2-bit input words into 4-bit codewords under the rules set by the FSSM. The source sequence is partitioned into 2-bit source words. Let a sample source sequence be '10 01 10 11', and assume an initial state $s_0 = \sigma_1$. Then the coded output sequence can easily be obtained. The next state is $s_1 = g(s_0 = \sigma_1, '10')$. In the same way: $s_2 = \sigma_2$, $s_3 = \sigma_3$, and $s_4 = \sigma_2$. The resulting output sequence is '1010 1001 1010 0101'.

Table 3.1: Codebook of three-state $R = 1/2$ code.

β	$h, g(\sigma_1, \beta)$	$h, g(\sigma_2, \beta)$	$h, g(\sigma_3, \beta)$
00	0110, σ_1	0110, σ_2	0110, σ_3
01	1001, σ_1	1001, σ_2	1001, σ_3
10	1010, σ_2	1010, σ_3	0011, σ_1
11	1100, σ_3	0101, σ_1	0101, σ_2

In the next section, we will continue the discussion of the spectra of encoded sequences.

3.5 Spectra of block-coded signals

Many codes that we will encounter in the next chapters are constituted by block codes. In this code format the source information is grouped in source words (blocks) of m bits. The m -bit source information is translated according to the encoding rules into blocks of n bits called codewords.

Let $\mathbf{x}_j = (x_{j,1}, \dots, x_{j,n})$ denote the j th transmitted codeword. The symbols $x_{j,i}$, $1, \leq i \leq n$, comprising the j th codeword are assumed to be sent serially. Then the general expression for a block-encoded sequence is

$$X(t) = \sum_{j=-\infty}^{\infty} \sum_{i=1}^n x_{j,i} s[t - (jn + i - 1)], \quad (3.37)$$

where as in Section 3.2.2, $s(t)$ denotes the standard pulse shape. We have seen in Section 3.2.2 that a time synchronous pulse amplitude signal is cyclostationary with the period equal to the channel bit interval. Block codes may exhibit a second periodicity, of period n , and the equivalent stationary statistics are obtained by phase-averaging over this longer period. The auto-correlation of the emitted sequence must, therefore, take into account the block interval and the position of both symbols within a codeword. Cariolaro & Tronca [48], Biglieri & Caire [30] introduced an elegant matrix formulation for the correlations, as follows. Define the $n \times n$ matrix

$$R_k = E\{\mathbf{x}_t^T \mathbf{x}_{t+k}\}, \quad k = 0, \mp 1, \mp 2, \dots, \quad (3.38)$$

where the expectation operator $E\{\cdot\}$ applies to all entries of the matrix. The equivalent phase-averaged process has auto-correlation function

$$nR_x(kn + l) = \sum_{i=1}^{n-l} [R_k]_{i,l+i} + \sum_{i=n-l+1}^n [R_{k+1}]_{i,l+i-n}, \quad 0 \leq l \leq n-1, \quad (3.39)$$

where $[R_k]_{u,v}$ denotes the u, v entry of R_k . Note that each distinct term in the series is the average of n contributions deriving from different symbol

positions, in accordance with the phase-averaging principle. The process of phase-averaging the auto-correlation function is visualized in Figure 3.2. For $l = 0$, the phase-averaged $R_x(kn + l)$, $k = 0, \mp 1, \mp 2, \dots$, is given by the arithmetic average of the diagonal entries of R_k , whereas for $l > 0$ we have to average some upper diagonal entries of R_k and some lower diagonal entries of R_{k+1} . The equivalent power spectrum is

$$H_x(\omega) = \sum_{i=-\infty}^{\infty} R_x(i) e^{-ji\omega}. \quad (3.40)$$

Let $\boldsymbol{\omega}$ be the row vector

$$\boldsymbol{\omega} = (e^{j\omega}, e^{j2\omega}, \dots, e^{jn\omega})$$

with transposed conjugate $\boldsymbol{\omega}^*$. Then $H_x(\omega)$ can be cast into the following elegant expression

$$H_x(\omega) = \frac{1}{n} \sum_{k=-\infty}^{\infty} \boldsymbol{\omega} R_k \boldsymbol{\omega}^* e^{-jkn\omega}. \quad (3.41)$$

It can be shown by writing out that this relation is identical with (3.40).

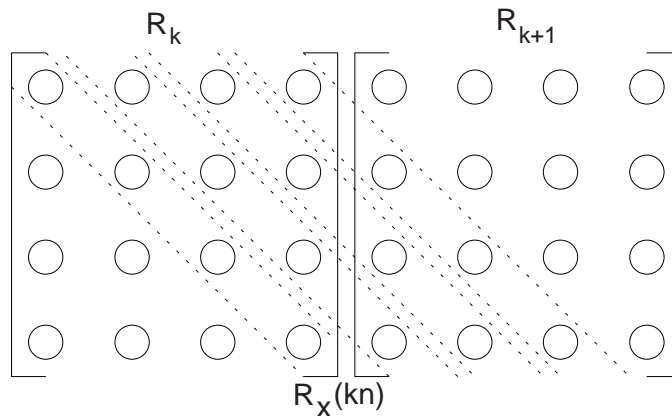


Figure 3.2: Visualization of the process of phase-averaging the auto-correlation function. Each distinct term in the series is the average of n contributions deriving from different symbol positions, in accordance with the phase-averaging principle. After Cariolaro *et al.* 1983 [49].

A general expression of the power spectral density function is

$$H_x(\omega) = H_{xc}(\omega) + H_{xd}(\omega) \sum_{k=-\infty}^{\infty} 2\pi\delta(\omega - 2\pi k/n), \quad (3.42)$$

where $H_{xc}(\omega)$ and $H_{xd}(\omega)$ designate the *continuous* and *discrete components* or *spectral lines* of the spectrum. The δ -functions in (3.42) represent power concentrated at various multiples of the codeword frequency $1/n$. Necessary and sufficient conditions for the encoded sequence to have a spectral line of given amplitude and frequency have been derived by Kamabe [184]. In general, the discrete components are considered to be a waste of power as far as information transmission is concerned. The presence of spectral lines at certain frequencies is not wholly undesirable in certain applications since they can be exploited to extract timing or position information. Other types of periodic behavior, such as that resulting from periodic insertion of synchronizing pulses, may also give rise to discrete components in the power spectral density function.

After rearrangement of (3.41), we can partition the spectrum $H_x(\omega)$ into discrete and continuous parts:

$$H_{xc}(\omega) = \frac{1}{n}\boldsymbol{\omega}(R_0 - R_\infty)\boldsymbol{\omega}^* + \frac{2}{n}\text{Re} \sum_{k=1}^{\infty} \boldsymbol{\omega}(R_k - R_\infty)\boldsymbol{\omega}^* e^{-jkn\omega} \quad (3.43)$$

and

$$H_{xd}(\omega) = \frac{1}{n^2}\boldsymbol{\omega}R_\infty\boldsymbol{\omega}^*. \quad (3.44)$$

R_∞ is the limit of the correlation matrices R_k as $k \rightarrow \infty$. The determination of the spectral density, then, requires the calculation of

1. The sequence of correlation matrices $R_k, k = 0, 1, 2, \dots$
2. Its limit

$$R_\infty = \lim_{k \rightarrow \infty} R_k. \quad (3.45)$$

3. The sum of the matrix series (3.43) and (3.44).

The above matrices in turn require the knowledge of the codeword bivariate probabilities, which are determined in the next subsection.

A great deal of the utility of the previous spectral computation would be lost if it were not possible to model an encoder as a finite-state (sequential) machine. It turns out, however, that most of the implemented encoder schemes can be simply treated in terms of Markov models. The next section provides a formal description of an encoder.

3.5.1 Spectral analysis of Moore-type encoders

Under general assumptions on the coding mechanism, it is possible to derive in closed form an expression for the power spectral density function of sequences produced. The computation of the spectrum of a block-encoded

signal is slightly more complicated than that of the spectrum of a sequence emitted by an ergodic Markov source [355].

An encoder controlled by a source of independent random inputs can be modelled by a Moore-type finite-state sequential machine (FSSM). It is characterized by an $N \times N$ transition probability matrix Q and the output matrix Γ of dimension $N \times n$, whose i th row is the codeword $\boldsymbol{\chi}_i = h(\sigma_i)$ of length n transmitted when the machine visits the state σ_i . The transition probabilities q_{ij} can be found by identifying the subset of input words that, on their occurrence of state i , cause a transition to state j . Let the stationary distribution be denoted by $\boldsymbol{\pi}$. In a Moore-type FSSM driven by a stationary source composed of independent and equiprobable symbols, the correlation matrix is given by

$$R_k = \Gamma^T \Pi Q^{|k|} \Gamma, \quad (3.46)$$

where Π is the diagonal matrix $\Pi = \text{diag}\{\pi_1, \dots, \pi_N\}$. This reduces, ignoring some notational convention, to (3.31) when $n = 1$. Throughout this text, the analysis is conducted on the assumption that the source words are independent and equiprobable. The theory can, however, be extended to the more general case where the source words are independent.

Now, from the theory of the ergodic and regular Markov chains Q has all its eigenvalues with modulus less than unity, with the exception of a simple eigenvalue $\lambda_1 = 1$. As a consequence, see [48], the limiting transition probability matrix exists such that

$$Q_\infty = \lim_{k \rightarrow \infty} Q^k = \mathbf{1}\boldsymbol{\pi},$$

that is, Q_∞ has all its rows equal to the steady-state probability vector $\boldsymbol{\pi}$. The codeword mean value correlation is given by

$$\mathbf{m} = \boldsymbol{\pi}\Gamma.$$

Passing to the limit in (3.46), we obtain

$$R_\infty = \lim_{k \rightarrow \infty} R_k = \Gamma^T \Pi Q_\infty \Gamma = \Gamma^T \boldsymbol{\pi}^T \boldsymbol{\pi} \Gamma = \mathbf{m}^T \mathbf{m}, \quad (3.47)$$

where use is made of the relation $\Pi Q_\infty = \Pi \mathbf{1}\boldsymbol{\pi} = \boldsymbol{\pi}^T \boldsymbol{\pi}$. We shall illustrate the preceding theory with an example which describes in detail the spectral properties of the code called MFM.

Example 3.2 MFM, (the acronym stands for Modified Frequency Modulation) also called *Delay Modulation*, is a rate 1/2 code with the virtue that at least two and at most four consecutive like symbols may occur in the transmitted sequence. This particular code was patented by Miller of Ampex in 1963 and is therefore often called the *Miller code* [247]. The MFM code belongs to the

important category of runlength-limited codes which will be described in detail in Chapter 5. As indicated by Hecht and Guida [128], the MFM coding scheme can be modelled as a Markov source with four states.

The encoding procedure can be described with the state-transition diagram shown in Figure 3.3. Perusal of Figure 3.3 leads to the following state-transition and codeword matrix:

$$Q_{\text{MFM}} = \begin{bmatrix} 0 & 1/2 & 0 & 1/2 \\ 0 & 0 & 1/2 & 1/2 \\ 1/2 & 1/2 & 0 & 0 \\ 1/2 & 0 & 1/2 & 0 \end{bmatrix}, \quad \Gamma = \begin{bmatrix} -1 & -1 \\ -1 & +1 \\ +1 & -1 \\ +1 & +1 \end{bmatrix}.$$

The limiting correlation matrix Q_{MFM}^{∞} can be found by inspection:

$$Q_{\text{MFM}}^{\infty} = \frac{1}{4} \begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 \end{bmatrix}$$

The stationary state probabilities are

$$\pi_1 = \pi_2 = \pi_3 = \pi_4 = \frac{1}{4}.$$

Thus

$$\Pi = \frac{1}{4} \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

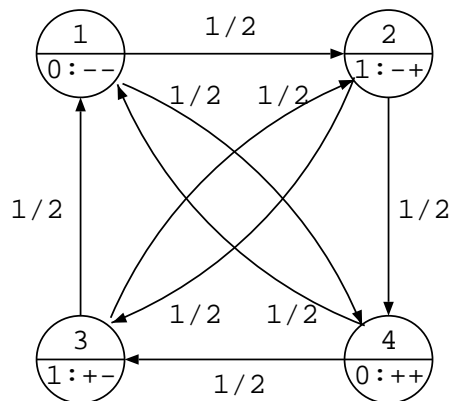


Figure 3.3: Moore-type state-transition diagram of MFM code. After Lindholm 1978 [226]. The four states, embodied by circles, are connected by arrows which represent allowed transitions from one state to the other. Along the edges we have indicated the probability that the chain goes from one state to the other. Within the circles we have indicated (a) the state number, (b) the source symbol, and (c) the waveform transmitted by the chain when it enters a certain state. For example, the word (+1, -1) is emitted when the chain enters State 3.

So that we conclude $\mathbf{m} = \mathbf{0}$ and $R_\infty = 0$. From the simple fact that $R_\infty = 0$, we infer that the power spectral density function of the MFM code has no spectral lines, or $H_{xd}(\omega) = 0$. After a computation we find

$$R_0 = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}, \quad R_1 = \frac{1}{2} \begin{bmatrix} -1 & -1 \\ 1 & -1 \end{bmatrix}$$

and

$$R_2 = \frac{1}{2} \begin{bmatrix} 0 & 1 \\ -1 & 0 \end{bmatrix}, \quad R_3 = \frac{1}{4} \begin{bmatrix} 1 & -1 \\ 1 & 1 \end{bmatrix}.$$

The following useful characteristic can be written down:

$$R_{k+4} = -\frac{1}{4}R_k, \quad k \geq 0. \quad (3.48)$$

After the phase averaging process, using (3.39), we obtain

$$\begin{aligned} R(2i) &= \frac{1}{2} \{ [R_i]_{1,1} + [R_i]_{2,2} \} \\ R(2i+1) &= \frac{1}{2} \{ [R_i]_{1,2} + [R_{i+1}]_{2,1} \}, \quad i \geq 0. \end{aligned} \quad (3.49)$$

Thus, the list of eight values of the auto-correlation function, derived after some bookkeeping, listed in Table 3.2 completely specifies $R(i)$.

Table 3.2: The correlation function $R(i)$ for MFM code.

$R(0) = 1$	$R(4) = 0$
$R(1) = 0.25$	$R(5) = 0.375$
$R(2) = -0.5$	$R(6) = 0.25$
$R(3) = -0.5$	$R(7) = -0.125$

The power spectral density function of MFM is

$$H_{\text{MFM}}(\omega) = 1 + 2\text{Re} \left[\frac{1}{1 + \frac{e^{-j8\omega}}{4}} \sum_{i=1}^8 R(i)e^{-ji\omega} \right].$$

After an evaluation, we obtain

$$H_{\text{MFM}}(\omega) = \frac{3 + \cos \omega + 2 \cos 2\omega - \cos 3\omega}{9 + 12 \cos 2\omega + 4 \cos 4\omega}.$$

The spectral density function of the MFM code is plotted in Figure 3.4. Since the curves are always even functions, only the base interval $0 \leq f = \omega/(2\pi) \leq 1/2$ is shown in the illustrations that follow.

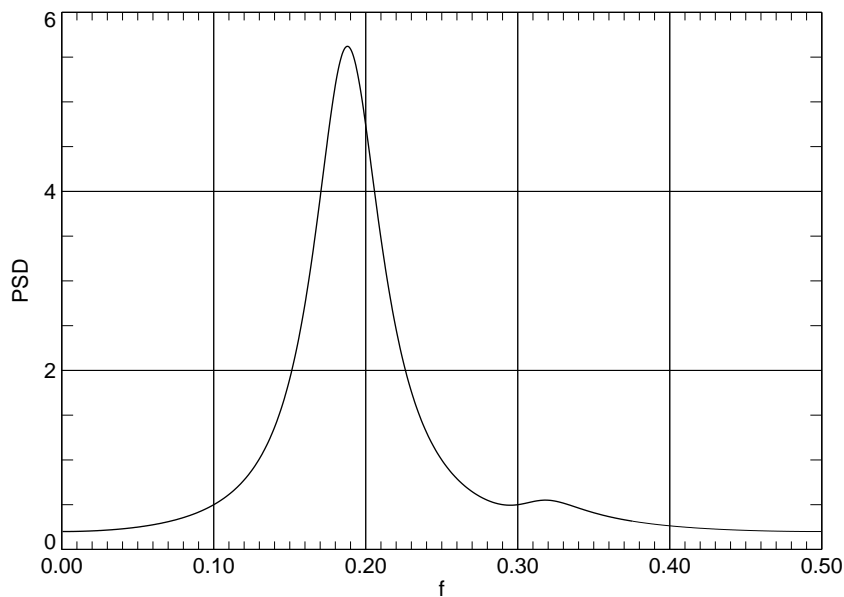


Figure 3.4: Power density function of MFM code as a function of the frequency $f = \omega/(2\pi)$.

3.5.2 Spectrum of memoryless block codes

Having discussed the general spectral analysis of block-coded sequences, we turn in this section to the relatively simple case of sequences generated by a memoryless encoder. In other words, we consider a type of encoder which has the property that there is an unambiguous, one-to-one mapping between source words and their associated codewords. This is a valid representative of some practical block codes, and this type of encoder model allows us to write down simplified relations of the power spectral density function of the generated sequence. The digression of the general model provides a simple interpretation, and it also serves to introduce some specific results that are used in the subsequent chapters.

To this end, assume the codeword set X comprises M codewords, the u th element of the codeword set is denoted by $\mathbf{x}_u = (\chi_1^{(u)}, \dots, \chi_n^{(u)})$, $0 \leq u \leq M - 1$. It is not required that M is a power of 2, albeit in many practical cases this is indeed the case. The codewords are randomly chosen from the codebook X to form an infinite sequence, which is sent serially. The power spectral density function of the emitted sequence can be divided again into a continuous and a discrete (line) part:

$$H(\omega) = H_c(\omega) + H_d(\omega) \sum_{k=-\infty}^{\infty} 2\pi\delta(\omega - 2\pi k/n), \quad (3.50)$$

where $H_c(\omega)$ and $H_d(\omega)$ denote the continuous and discrete components of the spectrum. The combination of (3.43), (3.44), (3.47), and using the fact that for a memoryless encoder $R_k = 0$, $k \neq 0$, leads to the following simplified equations:

$$H_c(\omega) = \frac{1}{n} \boldsymbol{\omega} (R_0 - R_\infty) \boldsymbol{\omega}^* \quad (3.51)$$

and

$$H_d(\omega) = \frac{1}{n^2} \boldsymbol{\omega} R_\infty \boldsymbol{\omega}^*. \quad (3.52)$$

After rearrangement of these equations, we obtain

$$H_d(\omega) = \frac{1}{n^2} \left\{ \sum_{i=1}^n \nu_i^2 + 2 \sum_{k=1}^{n-1} \sum_{i=1}^{n-k} \nu_i \nu_{i+k} \cos k\omega \right\} \quad (3.53)$$

and

$$H_c(\omega) = \frac{1}{n} \left\{ \mu_0 + 2 \sum_{k=1}^{n-1} \mu_k \cos k\omega \right\}, \quad (3.54)$$

where

$$\nu_k = \frac{1}{M} \sum_{u=0}^{M-1} \chi_k^{(u)}, \quad 1 \leq k \leq n \quad (3.55)$$

and

$$\mu_k = \frac{1}{M} \sum_{i=1}^{n-k} \sum_{u=0}^{M-1} (\chi_i^{(u)} - \nu_i)(\chi_{i+k}^{(u)} - \nu_{i+k}), \quad 0 \leq k \leq n-1. \quad (3.56)$$

As a direct consequence, we conclude that the spectrum is line free, provided $\nu_k = 0$, $k = 1, \dots, n-1$. Stated alternatively, the spectrum of a memoryless block code is line free if for all i , $1 \leq i \leq n$, the number of codewords $\boldsymbol{\chi}_u \in X$ with $\chi_i^{(u)} = 1$ equals the number of codewords with $\chi_i^{(u)} = -1$ (i.e. the sum of all codewords in X is the all-zero vector). The non-existence of a line spectrum is guaranteed (sufficient condition) when the code set consists of codeword pairs of opposite polarity. It should be appreciated that the preceding outcomes hold only (this has been the assumption throughout this chapter) provided all words are equiprobable.

We next consider an alternative expression for the power spectral density function of a memoryless block code which is chosen for its simplicity and ease of interpretation. For clerical convenience we confine ourselves to the assumption that the spectrum is line free, i.e. $H_d(\omega) = 0$. Let $\boldsymbol{\chi}_u = (\chi_1^{(u)}, \dots, \chi_n^{(u)})$ be the u th element of a set X of codewords. The Fourier transform $X^{(u)}(\omega)$ of the codeword $\boldsymbol{\chi}_u$ is defined by

$$X^{(u)}(\omega) = \sum_{i=1}^n \chi_i^{(u)} e^{-ji\omega}. \quad (3.57)$$

If, again, codewords are randomly chosen from the codebook X to form an infinite sequence, then it is straightforward to show that the power spectral density function $H(\omega)$ of the concatenated sequence, when the symbols are transmitted serially, is given by

$$H(\omega) = \frac{1}{Mn} \sum_{u=0}^{M-1} |X^{(u)}(\omega)|^2. \quad (3.58)$$

It can be shown by multiplying out the matrix products that the preceding equation is equivalent with (3.51). Equation (3.58) reveals that the power spectral density function of the cascaded sequence is the average of the spectra of each individual codeword. In the next example we shall elaborate on the preceding theory, and establish the power spectral density function of a sequence generated by a memoryless encoder which employs a set of codewords with equal numbers of 'ones' and 'zeros'.

Example 3.3 Assume the codeword set comprises two codewords, namely

$$\chi_1 = (+1, -1)$$

and its inverse

$$\chi_2 = (-1, +1).$$

This block code, called *bi-phase*, also referred to as *Manchester code*, is a popular rate 1/2, encoding function in low-end magnetic disk drives. The reasons for the popularity are self-clocking capability and extremely simple encoding and decoding circuits. The spectrum is line free as $\chi_1 = -\chi_2$. The Fourier transform of the codewords is

$$X^{(1)}(\omega) = e^{-j\omega} - e^{-2j\omega}$$

and

$$X^{(2)}(\omega) = -e^{-j\omega} + e^{-2j\omega}.$$

So that, using (3.58), the spectrum of the bi-phase code, $H_{\text{Bi}\Phi}(\omega)$, is

$$H_{\text{Bi}\Phi}(\omega) = \frac{1}{4} \left\{ |X^{(1)}(\omega)|^2 + |X^{(2)}(\omega)|^2 \right\} = 2 \sin^2 \frac{\omega}{2}.$$

Note that $H_{\text{Bi}\Phi}(0) = 0$, which is the reason why this code is called a *dc-balanced* or *dc-free code*. In the next example we shall elaborate on the preceding example, and establish the power spectral density function of more dc-free code comprising a set of codewords with equal numbers of 'ones' and 'zeros'.

Example 3.4 Let X consist of all sequences $\chi = (\chi_1, \chi_2, \dots, \chi_n) \in \{-1, 1\}^n$ such that $\sum_{i=1}^n \chi_i = 0$. A codeword with equal numbers of +1s and -1s is called a *zero-disparity* codeword and, obviously, the codeword length n is even. The number of zero-disparity codewords, M , is given by the binomial coefficient

$$M = |X| = \binom{n}{n/2}. \quad (3.59)$$

By virtue of the fact that the codeword set consists of pairs of codewords of opposite polarity, we conclude that the spectrum is line free, or $H_d(\omega) = 0$.

For symmetry reasons we conclude that the correlation between symbols does not depend on the actual symbol positions j_1 and j_2 , $j_1 \neq j_2$, within a codeword, or

$$\frac{1}{M} \sum_{u=0}^{M-1} \chi_{j_1}^{(u)} \chi_{j_2}^{(u)} = \varepsilon, \quad j_1 \neq j_2.$$

Using (3.56) yields

$$\mu_k = \frac{1}{M} \sum_{i=1}^{n-k} \sum_{u=0}^{M-1} \chi_i^{(u)} \chi_{i+k}^{(u)} = (n-k)\varepsilon, \quad 1 \leq k \leq n-1.$$

As $\chi_i \in \{-1, 1\}$, we have

$$\begin{aligned} \varepsilon &= \frac{1}{M} \sum_{u=0}^{M-1} \chi_{j_1}^{(u)} \chi_{j_2}^{(u)} = \frac{1}{M} \{N(\chi_{j_1} \chi_{j_2} = 1) - N(\chi_{j_1} \chi_{j_2} = -1)\} \\ &= \frac{2}{M} N(\chi_{j_1} = \chi_{j_2}) - 1 \\ &= \frac{4}{M} N(\chi_{j_1} = \chi_{j_2} = 1) - 1, \quad j_1 \neq j_2, \end{aligned} \tag{3.60}$$

where $N(\cdot)$ denotes the number of codewords satisfying the condition in parentheses. By inspection, we find that the number of codewords for which $\chi_{j_1} = \chi_{j_2} = 1$ (it is simply the number of words of length $n-2$ with $n/2$ symbols equal to -1) equals

$$N(\chi_{j_1} = \chi_{j_2} = 1) = \binom{n-2}{n/2},$$

or

$$\frac{4}{M} N(\chi_{j_1} = \chi_{j_2} = 1) = 4 \binom{n-2}{n/2} / \binom{n}{n/2} = \frac{n-2}{n-1},$$

so that

$$\mu_k = \begin{cases} n, & k = 0, \\ \frac{k-n}{n-1}, & 0 < k \leq n, \\ 0, & k > n. \end{cases} \tag{3.61}$$

After a routine computation, using (3.50), (3.54), and (3.61), we can write down the power spectral density

$$H_n(\omega) = \frac{n}{n-1} \left\{ 1 - \left(\frac{\sin n\omega/2}{n \sin \omega/2} \right)^2 \right\}. \tag{3.62}$$

Note from the above equation that $H_n(0) = 0$, that is the power vanishes at the zero frequency, which is the reason why this group of codes is usually called *dc-free* or *dc-balanced code*. Dc-free codes are of great weight for input-constrained channels. Many other examples of dc-free codes will be given in the next chapters. Figure 3.5 shows the power density function, $H_n(\omega)$, of zero-disparity codes versus frequency with the codeword length, n , as a parameter. Let the codeword length

be $n = 2$. In this elementary case, there are only two zero-disparity codewords, namely $(1, -1)$ and its inverse $(-1, 1)$, and the power spectral density function $H_2(\omega)$ of the bi-phase code can be readily derived to be equal to the result as previously derived in Example 3.3.

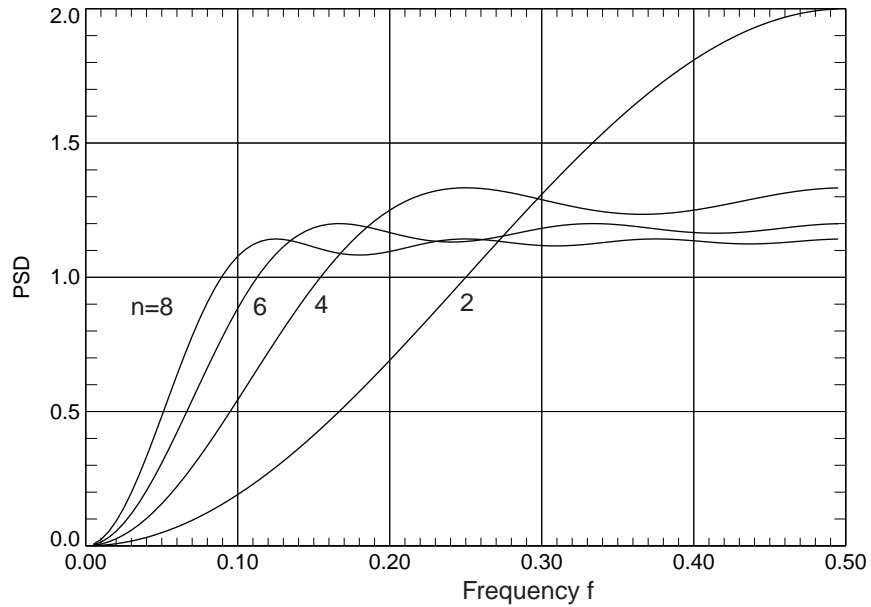


Figure 3.5: Power density function of zero-disparity codes versus frequency, $f = \omega/2\pi$, with the codeword length n as a parameter. The power vanishes at the zero frequency which is the reason why this group of codes is usually called dc-free or dc-balanced.

Chapter 4

Runlength-limited Sequences: Theory

4.1 Introduction

Codes based on runlength-limited sequences have been the state of the art corner stone of current disc recorders whether their nature is magnetic or optical. In this chapter, we shall provide a detailed description of various properties of runlength-limited sequences and in the next chapter we will give a comprehensive review of the code construction methods, ad hoc as well as systematic, that are available.

The length of time usually expressed in channel bits between consecutive transitions is known as the *runlength*. For instance, the runlengths in the word '0111100111000000' are of length 1, 4, 2, 3, and 6. Runlength-limited (RLL) sequences are characterized by two parameters, $(d + 1)$ and $(k + 1)$, which stipulate the minimum (with the exception of the very first and last runlength) and maximum runlength, respectively, that may occur in the sequence. The parameter d controls the highest transition frequency and thus has a bearing on intersymbol interference when the sequence is transmitted over a bandwidth-limited channel. In the transmission of binary data it is generally desirable that the received signal is self-synchronizing or self-clocking. Timing is commonly recovered with a phase-locked loop which adjusts the phase of the detection instant according to observed transitions of the received waveform. The maximum runlength parameter k ensures adequate frequency of transitions for synchronization of the read clock. The grounds on which d and k values are chosen, in turn, depend on various factors such as the channel response, the desired data rate (or information density), and the jitter and noise characteristics.

Recording codes that are based on RLL sequences have found almost universal application in disc recording practice. In consumer electronics, we have the EFM code (rate = 8/17, $d = 2$, $k = 10$), which is employed in the

Compact Disc (CD), and the EFMPlus code (rate = 8/16, $d = 2$, $k = 10$) used in the DVD, the successor of the CD. The future of RLL sequences is very bright as also in new mass storage products, such as BluRay Disc, a rate 2/3, parity preserving word assignment (see Section 11.4.3, page 288) (1,7) dc-free RLL code has been adopted [260].

Runlength-limited codes, in their general form, were pioneered in the 1960s by Berkoff [27], Freiman & Wyner [101], Kautz [196], Gabor [106], Shaft [297], Tang & Bahl [319, 320], and notably Franaszek [96]. It is undoubtedly the case that RLL codes have generated a high and sustained level of interest amongst researchers ever since the introduction of the basic ideas. There is now a considerable amount of literature available on the design of encoding and decoding devices for generating RLL sequences. Before detailing the theory of RLL sequences, it is convenient to introduce 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 'one's are separated by a run of consecutive 'zero's of length at least d .
2. k constraint - any run of consecutive 'zero's is of length at most k .

If only proviso (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 coding step. A (dk) sequence is converted to a runlength-limited channel sequence in the following way. Let the channel signals be represented by a bipolar 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 'one's in the (dk) sequence indicate the positions of a transition $1 \rightarrow -1$ or $-1 \rightarrow 1$ of the corresponding RLL sequence. The (dk) sequence

$$0\ 1\ 0\ 0\ 0\ 1\ 0\ 0\ 1\ 0\ 0\ 0\ 1\ 1\ 0\ 1\ \dots$$

would be converted to the RLL channel sequence

$$1\ -1\ -1\ -1\ -1\ 1\ 1\ 1\ -1\ -1\ -1\ -1\ 1\ -1\ -1\ 1\ \dots$$

The mapping of the waveform by this coding step is known as *precoding*. A confusing term since it is in fact a 'postcoding process'. We shall avoid this term, and use the designation *change-of-state* encoding instead. Sequences that are assumed to be recorded with such a change-of-state encoding step,

as for example, (dk) sequences, are said to be given in non-return-to-zero-inverse NRZI¹ notation. Waveforms that are transmitted without such an intermediate coding step are referred to as non-return-to-zero (NRZ). The names stem from telegraphy and have no meaning in relation to recording channels. These nebulous terms are in common use and we will continue its use in the rest of this text. Coding techniques using the NRZI format are generally accepted in digital optical and magnetic recording practice. The NRZI format is convenient in magnetic recording since differentiation occurs as part of the physical process in the heads. The original signal is restored in a quite natural fashion by observing the peaks in the retrieved signal. The peaks coincide with the 'one's of the stored sequence in NRZI notation. The use of the NRZI format in non-differentiating channels, such as the Compact Disc, is less obvious.

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 (runs) occur.

Table 4.1: Various codes with runlength parameters d and k .

d	k	R	
0	1	1/2	FM, Bi-phase
1	3	1/2	MFm, Miller
2	7	1/2	(2,7)
1	7	2/3	(1,7)
2	11	1/2	3PM
2	10	8/17	EFM

In Table 4.1 we have collected some parameters of runlength-limited codes that have found practical application. The characteristics of the various codes will be explained in this and other chapters to come.

The outline of this chapter is as follows. Section 4.2 addresses the problem of counting the number of RLL sequences of a given length. Thereafter we compute the asymptotic information rate, capacity, of RLL sequences and then turn to the description of the statistical characteristics of maxentropic RLL sequences. Of particular note is Section 4.4, which deals with the properties of maxentropic RLL sequences. The remaining sections deal with the properties of sequences with a variety of runlength constraints de-

¹There is no consensus on the meaning of the letter 'I'. The origin of the invention in relation to magnetic recording seems to be Phelps [283].

veloped to counteract all kind of physical imperfections of the recording channel. In the final section, we will discuss two-dimensional RLL codes.

4.2 Counting (dk) sequences

In this section we address the problem of counting the number of sequences of a certain length which comply with given dk constraints. We start for the sake of clerical convenience with the enumeration of (d) sequences. Let $N_d(n)$ denote the number of distinct (d) sequences of length n and define

$$\begin{aligned} N_d(n) &= 0, \quad n < 0, \\ N_d(0) &= 1. \end{aligned} \tag{4.1}$$

The number of (d) sequences of length $n > 0$ is found with the recursive relations [319]

$$\begin{aligned} (i) \quad N_d(n) &= n + 1, \quad 1 \leq n \leq d + 1, \\ (ii) \quad N_d(n) &= N_d(n - 1) + N_d(n - d - 1), \quad n > d + 1. \end{aligned} \tag{4.2}$$

The proof of (4.2), taken from [319], is straightforward.

- i. If $n \leq d + 1$, a (d) sequence can contain only a single 'one' (and there are exactly n such sequences), or the sequence must be the all 'zero' sequence (and there is only one such sequence).
- ii. If $n > d + 1$, a (d) sequence can be built by one of the following procedures:
 - To build any (d) sequence of length n starting with a 'zero', take the concatenation of a 'zero' and any (d) sequence of length $n - 1$. There are $N_d(n - 1)$ of such.
 - Any (d) sequence of length n starting with a 'one' can be constructed by the concatenation of a 'one' and d 'zero's followed by any (d) sequence of length $n - d - 1$. There are $N_d(n - d - 1)$ of such.

Table 4.2 lists the number of distinct (d) sequences as a function of the sequence length n with the minimum runlength d as a parameter. 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

$$1, 2, 3, 5, 8, 13, \dots,$$

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 [331]. For $d > 1$, the numbers $N_d(n)$ are often called *generalized Fibonacci numbers*.

Table 4.2: Number of distinct (d) sequences as a function of the sequence length n and the minimum runlength d as a parameter.

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

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

$$\begin{aligned} N(n) &= 0, \quad n < 0, \\ N(0) &= 1. \end{aligned} \tag{4.3}$$

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

$$\begin{aligned} N(n) &= n + 1, \quad 1 \leq n \leq d + 1, \\ N(n) &= N(n - 1) + N(n - d - 1), \quad d + 1 \leq n \leq k, \\ N(n) &= d + k + 1 - n + \sum_{i=d}^k N(n - i - 1), \quad k < n \leq d + k, \\ N(n) &= \sum_{i=d}^k N(n - i - 1), \quad n > d + k. \end{aligned} \tag{4.4}$$

The proof of the above recursion relations is not interesting and therefore omitted [319]. The k -limited case, $d = 0$, can be derived as a special case of the general dk case. If $N_k(n)$ denotes the number of (k) sequences of length n , the following recursion relations can be written down:

$$\begin{aligned} N_k(n) &= 2^n, \quad 0 < n \leq k, \\ N_k(n) &= \sum_{i=1}^{k+1} N_k(n - i), \quad n > k. \end{aligned} \tag{4.5}$$

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

4.3 Asymptotic information rate

An encoder translates arbitrary user (or source) information into, in this particular instance, a sequence that satisfies given (dk) 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, as discussed in Chapter 2, was given by Shannon [296]. The maximum value of R that can be achieved is called the *capacity*. The capacity, or asymptotic information rate, of (dk) sequences, denoted by $C(d, k)$, defined as the number of information bits per channel bit that can maximally be carried by the (dk) sequences, on average, is governed by the specified constraints and is given by

$$C(d, k) = \lim_{n \rightarrow \infty} \frac{1}{n} \log_2 N_{dk}(n). \quad (4.6)$$

For notational convenience we restrict ourselves for the time being to (d) sequences. Note that the capacity formula (4.6) requires an explicit formula of the number of sequences $N_d(n)$ as a function of the sequence length n . The desired expression is most easily obtained by solving the homogeneous *difference* equation (4.2). According to (4.2) the number of (d) sequences is

$$N_d(n) = N_d(n-1) + N_d(n-d-1), \quad n > d+1. \quad (4.7)$$

We assume a solution of the form

$$N_d(n) = cz^n,$$

where $c \neq 0$. Substituting this expression into (4.7) results in

$$cz^n(1 - z^{-1} - z^{-d-1}) = 0.$$

We are interested in a non-trivial solution, $N_d(n) \neq 0$, and consequently, z must be a root of the equation

$$1 - z^{-1} - z^{-d-1} = 0,$$

or, equivalently,

$$z^{d+1} - z^d = 1. \quad (4.8)$$

This equation is usually referred to as the characteristic equation. Any z that satisfies the characteristic equation will solve the difference equation. For the case where (4.8) has $d+1$ distinct roots λ_i , $i = 1, \dots, d+1$, it can be easily shown that from the linearity of (4.7) that the most general solution for $N_d(n)$ is of the form

$$N_d(n) = \sum_{i=1}^{d+1} a_i \lambda_i^n, \quad (4.9)$$

where a_i are constants, independent of n , to be chosen to meet the first $(d + 1)$ values of $N_d(n)$. If $\lambda = \max\{\lambda_i\}$ is the largest (positive) real root of (4.8), then for large values of n , (4.9) reduces to

$$N_d(n) \propto \lambda^n, \quad (4.10)$$

since the other terms in (4.9) become negligible by comparison. In (4.10), we have thus found the general solution of (4.2) for sufficiently large values of n and we infer that the number of possible (d) sequences grows exponentially with n , when n is large. Applying definition (4.6), the asymptotic information rate (or capacity) of d -constrained sequences, denoted by $C(d, \infty)$, is

$$C(d, \infty) = \lim_{n \rightarrow \infty} \frac{1}{n} \log_2 N_d(n) = \log_2 \lambda. \quad (4.11)$$

This is the fundamental result that we need: The quantity $C(d, \infty)$ provides the maximum rate possible of any implemented code given the minimum runlength constraint d .

Example 4.1 Let d equal 1, then we obtain the characteristic equation

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

with solutions

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

The general expression for $N_1(n)$ is then

$$N_1(n) = a_1 \left(\frac{1 + \sqrt{5}}{2} \right)^n + a_2 \left(\frac{1 - \sqrt{5}}{2} \right)^n, \quad n \geq 0. \quad (4.12)$$

After some rearrangement, using the initial conditions $N_1(0) = 1$ and $N_1(1) = 2$, we obtain, surprisingly, an explicit formula:

$$\begin{aligned} N_1(n) &= \frac{1}{\sqrt{5}} \left(\frac{1 + \sqrt{5}}{2} \right)^{n+2} - \frac{1}{\sqrt{5}} \left(\frac{1 - \sqrt{5}}{2} \right)^{n+2} \\ &= \frac{1}{\sqrt{5}} \{g^{n+2} - (-g)^{-n-2}\}, \quad n \geq 0. \end{aligned} \quad (4.13)$$

The above formula, discovered by de Moivre in 1718 and proved some years later by Bernoulli, can be used to determine any Fibonacci number without having to compute any other number in the Fibonacci sequence. The growth factor is $\lambda = g = (1 + \sqrt{5})/2$, and the capacity is

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

Some further outcomes of computations, which are obtained by numerical methods, are collected in Table 4.3.

Table 4.3: Capacity and density ratio T_{\min} 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

The quantity T_{\min} , called *density ratio*, or *packing density*, is defined as

$$T_{\min} = (1 + d)C(d, \infty). \quad (4.14)$$

The density ratio expresses the minimum physical distance between consecutive transitions of an RLL sequence given the information content is fixed. It can be seen from Table 4.3 that an increase of the density ratio is obtained at the expense of decreased capacity $C(d, k)$. The minimum increment between any physical runlength is called the *timing window* or *detection window* denoted by T_w . Clearly, $T_w = C(d, k)$. We conclude that sequences with a larger value of d , and thus a lower capacity $C(d, k)$, are penalized by an increasingly difficult trade-off between the detection window T_w and density ratio T_{\min} .

By rewriting (4.8), we obtain an interesting upper bound to T_w given a T_{\min} . By definition we have $\lambda = 2^{C(d, \infty)}$, so that using (4.8), we have

$$2^{-T_w} + 2^{-T_{\min}} = 1. \quad (4.15)$$

The above relationship is plotted in Figure 4.1. As the rate of an implemented code, R , is less or equal $C(d, \infty)$ the curve shows an upper bound to $T_{\min} = R(d + 1)$ of any implemented RLL code. Only discrete points on this curve are possible with maxentropic (d) constrained sequences. With so-called "RLL sequences with multiple spacings", see Section 4.5.4, more points on the curve are made possible. From (4.15) we easily derive for large values of T_{\min}

$$T_w \ln 2 \approx 2^{-T_{\min}}.$$

In similar vein to the case of d -constrained sequences, it is possible to derive the capacity $C(d, k)$ of (dk) sequences.

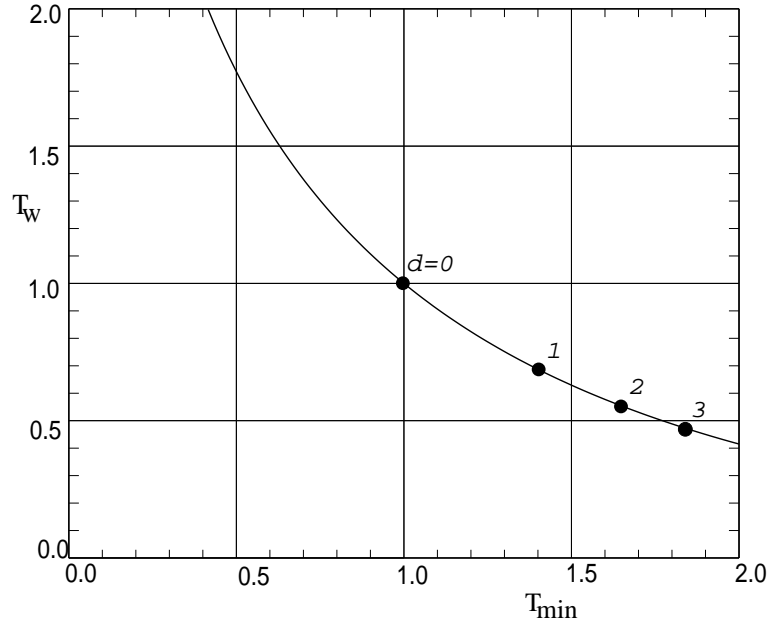


Figure 4.1: Bound to T_{\min} and window T_w . The circles indicate points pertaining to (d) sequences.

Invoking recursion relation (4.4), we can write down the characteristic equation

$$z^{-(k+1)} + z^{-k} + \dots + z^{-(d+1)} - 1 = 0,$$

or equivalently

$$z^{k+2} - z^{k+1} - z^{k-d+1} + 1 = 0. \quad (4.16)$$

Alternatively, the theory developed in Chapter 2 can be used to derive the capacity $C(d, k)$ of (dk) sequences. Sequences that meet prescribed (dk) constraints may be thought to be composed of *phrases* of length (duration) $j + 1$, $d \leq j \leq k$, denoted by T_{j+1} , from the set of phrases

$$\{10^d, 10^{d+1}, \dots, 10^j, \dots, 10^k\},$$

where 0^j stands for a sequence of j consecutive 'zero's. As an immediate consequence of (2.27), page 25, the characteristic equation of (dk) sequences is (for finite k)

$$z^{-(k+1)} + z^{-k} + \dots + z^{-(d+1)} - 1 = 0,$$

which is similar to the characteristic equation derived above. The characteristic equation of k -constrained sequences is immediate:

$$z^{k+2} - 2z^{k+1} + 1 = 0 \quad (4.17)$$

We derive for sufficiently large k

$$\lambda \simeq 2\left(1 - \frac{1}{2^{k+2}}\right)$$

and

$$C(0, k) \simeq 1 - \frac{1}{4 \ln 2} 2^{-k}, \quad k \gg 1.$$

Table 4.4 lists the capacity $C(d, k)$ versus the parameters d and k .

Table 4.4: Capacity $C(d, k)$ versus runlength parameters d and k .

k	$d = 0$	$d = 1$	$d = 2$	$d = 3$	$d = 4$	$d = 5$	$d = 6$
1	.6942						
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

4.3.1 State-transition matrix description

There is an alternative useful technique to derive the channel capacity, which is based on the representation of the (dk) constraints by a finite-state sequential machine. Figure 4.2 illustrates a possible state-transition diagram. There are $(k + 1)$ states which are denoted by $\{\sigma_1, \dots, \sigma_{k+1}\}$. 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 $\sigma_{d+1}, \dots, \sigma_{k+1}$. Any path through the state-transition diagram defines an allowed (dk) sequence. If r 'zero's has been transmitted since the last 'one', the machine is in state $r + 1$. The adjacency, or connection, matrix, which gives the

number of ways 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

$$\begin{aligned} d_{i1} &= 1, \quad i \geq d+1, \\ d_{ij} &= 1, \quad j = i+1, \\ d_{ij} &= 0, \quad \text{otherwise.} \end{aligned} \tag{4.18}$$

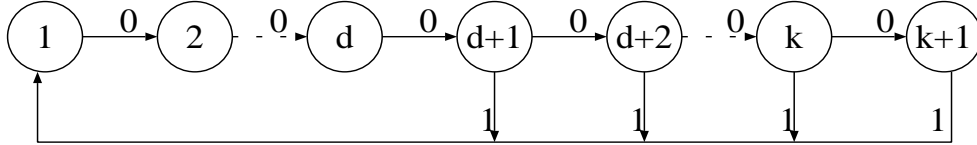


Figure 4.2: State-transition diagram for a (dk) sequence. Transmission of a 'zero' takes the sequence from state σ_i to state σ_{i+1} , $i \leq k$. A 'one' may be transmitted only when the machine occupies states $\sigma_{d+1}, \dots, \sigma_{k+1}$, while a 'one' must be produced if the machine is in state σ_{k+1} . The machine returns to state σ_1 after transition of a 'one'.

As an illustration, we have written down the connection matrix for the $(d, k) = (1, 3)$ constraints:

$$D = \begin{bmatrix} 0 & 1 & 0 & 0 \\ 1 & 0 & 1 & 0 \\ 1 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 \end{bmatrix}. \tag{4.19}$$

A d -constrained channel can be modelled with $d+1$ states (see Figure 4.3.) The connection matrix is given by the $(d+1) \times (d+1)$ array D with entries d_{ij} , where

$$\begin{aligned} d_{ij} &= 1, \quad j = i+1, \\ d_{d+1,1} &= d_{d+1,d+1} = 1, \\ d_{ij} &= 0, \quad \text{otherwise.} \end{aligned} \tag{4.20}$$

The above representation is an example of the input-restricted noiseless channel studied by Shannon (see Chapter 2). As shown in Chapter 2, the finite-state machine model allows us to compute the capacity, and it is also very helpful to compute the number of sequences that start and end in given states. According to Theorem 2.1, page 22, the n -step state-transition matrix has ij entries that give the number of distinct sequences from state i to state j that are n bits long.

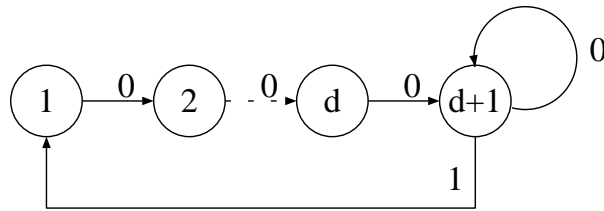


Figure 4.3: State-transition diagram for a (d) sequence. Only in state σ_{d+1} either a 'zero' or a 'one' may be emitted, in all other states a 'zero' must be emitted.

Applying (2.19), page 22, we find that the capacity of a Markov source that emits dk -constrained sequences is

$$C(d, k) = \log_2 \lambda,$$

where λ is the largest real root of the characteristic equation

$$\det[D - zI] = 0, \quad (4.21)$$

and I is the identity matrix. It is left for the reader as an exercise to demonstrate that the latter equation coincides with (4.16), page 59.

4.3.2 Useful properties

It has been shown by Ashley & Siegel [13] that, save a trivial exception, the capacity of binary (d, k) -constrained sequences is irrational. A similar result was obtained by McLaughlin & Xie [242] for M -ary (d, k) -constrained sequences. It is therefore 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.

Theorem 4.1 *The capacity $C(d, k)$ is irrational unless $d = 0$ and $k = \infty$.*

Proof: If x is a root of the equation

$$x^q + c_1 x^{q-1} + \cdots + c_m = 0,$$

with integral coefficients, c_i , then x is either integral or irrational [122]. As the maximum real root λ of (4.16) has $1 < \lambda < 2$, unless $d = 0$ and $k = \infty$, we conclude that λ is irrational. It can be seen by inspection that λ is not an r -root of two, and we therefore conclude that $C(d, k)$ is irrational. We conclude that codes can only be implemented with a rate, a rational number, which is smaller than (and not equal to) the Shannon capacity. As an example we have computed quotients $R = m/n < C(d, k)$, m and n integers, with decreasing relative distance to the Shannon capacity. Results

are given in Table 4.5. It can be seen that relatively small codes can, in principle, be constructed that can attain the Shannon capacity within a few tenths of a percent. Whether and how this can be done in practice, will be discussed in the various chapters to follow.

Table 4.5: Examples of rates of RLL codes, $R < C(d, k)$, as a function of the runlength parameters d and k .

d	k	R	$1 - R/C$	R	$1 - R/C$	R	$1 - R/C$
1	7	2/3	0.01858	19/28	0.00105	36/53	0.00006
2	7	1/2	0.03357	15/29	0.00025		
3	12	3/7	0.05921	4/9	0.02437	46/101	0.00022
5	12	1/3	0.01065	32/95	0.00024	63/187	0.00007

Ashley & Siegel [13, 10] derived a useful relation between the capacity of (dk) sequences and (d) sequences, namely

$$C(d, \infty) = C(d - 1, 2d - 1), \quad d \geq 1. \quad (4.22)$$

To prove (4.22) is not difficult. The characteristic equations of d -constrained and dk -constrained sequences are (see (4.8) and (4.16))

$$z^{d+1} - z^d - 1 = 0,$$

and

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

By substituting $d - 1$ for d and $2d - 1$ for k in the latter equation we obtain the characteristic equation

$$z^{2d+1} - z^{2d} - z^{d+1} + 1 = (z^d - 1)(z^{d+1} - z^d - 1) = 0.$$

The factor $(z^d - 1)$ has all its zeros on the unit circle. Therefore, we may state that both equations have the same largest real root, which concludes the proof of (4.22).

There is a simple relation between $(0,1)$ and $(1, \infty)$ sequences. Let $\{x_i\} = \{x_1, x_2, \dots\}$ be a $(0,1)$ sequence. Logical inversion of all symbols of $\{x_i\}$, i.e. $y_i = \bar{x}_i$, yields the sequence $\{y_i\}$ which, as can readily be seen, is a $(1, \infty)$ sequence. A second relationship was found by Forsberg & Blake [86]

$$C(d, 2d) = C(d + 1, 3d + 1). \quad (4.23)$$

The proof of the foregoing relationship (4.23) proceeds along the same lines as the proof of (4.22), and is therefore omitted.

Kashyap and Siegel [192] showed that repeatedly applying (4.22) and (4.23) yields the chain of equalities

$$C(1, 2) = C(2, 4) = C(3, 7) = C(4, \infty). \quad (4.24)$$

They showed that no equalities other than those listed in (4.22)-(4.24) are possible among the capacities $C(d, k)$. Given a pair of dk -constrained systems of the same capacity, a question that naturally arises is whether there exists a lossless finite-state encoder between the two dk -constrained systems that is sliding-block decodable, i.e., can be decoded using finite memory. The trivial translation between a $(0,1)$ and $(1,0)$ -constrained systems and vice versa has been discussed above. Kashyap and Siegel showed that a lossless translation from $C(d, 2d)$ to $C(d + 1, 3d + 1)$ -constrained systems for $d \geq 1$, is also possible. They also showed the nonexistence of certain translations.

4.4 Maxentropic RLL sequences

To compute the statistical properties, such as runlength distribution, power spectral density function, and so on, of sequences generated by implemented codes can be a difficult task. It is, however, relatively simple, using the theory developed in Chapter 2, to find the statistics of sequences emitted by an ideal or maxentropic source. The output of an encoder working at a rate close to channel capacity should, in some sense, approximate the maxentropic Markov chain, which justifies the fact that the outcomes based on maxentropic sequences may be used to approximate the characteristics of implemented codes.

In a maxentropic RLL sequence, the runlength of length T_i has probability of occurrence [146, 356, 357]

$$Pr(T_i) = \lambda^{-i}, \quad i = d + 1, \dots, k + 1, \quad (4.25)$$

where λ is the largest real root of the characteristic equation (4.16). For the proof of (4.25), we follow Howell [146] and Zehavi & Wolf [357].

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

$$D\hat{\mathbf{v}} = \lambda\hat{\mathbf{v}}. \quad (4.26)$$

The right eigenvector is $\hat{\mathbf{v}} =$

$$(1, \lambda, \lambda^2, \dots, \lambda^d, (\lambda^{d+1} - 1), (\lambda^{d+2} - \lambda - 1), \dots, (\lambda^k - \lambda^{k-d-1} - \dots - \lambda - 1))^T.$$

The joint probability q_{ij} of a transition from state i to j of the maxentropic Markov source is

$$q_{ij} = \frac{1}{\lambda} d_{ij} \frac{\hat{v}_j}{\hat{v}_i}, \quad i, j = 1, 2, \dots, k + 1. \quad (4.27)$$

The matrix Q is stochastic, that is $\sum_i q_{ij} = 1$. The probability π_i associated with state σ_i is given by $\hat{v}_i \hat{u}_i$, where $\hat{\mathbf{v}}$ and $\hat{\mathbf{u}}$ are the right and left eigenvectors for D with eigenvalue λ , respectively, and $\sum \pi_i = 1$. The left eigenvector $\hat{\mathbf{u}}$, is given by

$$\hat{\mathbf{u}} = (\lambda^k, \lambda^{k-1}, \dots, 1), \quad (4.28)$$

and the stationary probability π_i is

$$\rho \pi_i = \begin{cases} \lambda^k, & 1 \leq i \leq d + 1 \\ \lambda^k - \sum_{j=1}^{i-1-d} \lambda^{k-d-j} & d + 2 \leq i \leq k + 1 \end{cases} \quad (4.29)$$

where the normalization constant ρ is chosen to satisfy

$$\sum \pi_i = 1.$$

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

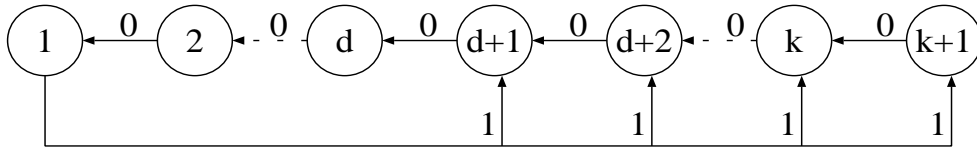


Figure 4.4: Reversed state-transition diagram. The figure is the same as Figure 4.2, but now the arrows are pointing in the opposite direction.

The reversed state-transition diagram describes the same set of runlength parameters as the original, the reverse transition diagram is shown in Figure 4.4. The reversed diagram affords a simpler way to compute the runlength distribution of a maxentropic RLL sequence than the original diagram. The probability of runlength g , $d \leq g \leq k$, 'zero's is just the probability \tilde{q}_{1g} from state σ_1 to state σ_g . Using the formulas $\tilde{q}_{ij} = \frac{1}{\lambda} \hat{u}_j / \hat{u}_i$ and $\hat{u}_i = \lambda^{k-i+1}$ from above, we obtain $\tilde{q}_{1g} = \lambda^{-g}$. The probability of a phrase of length g consisting of a 'one' followed by $g - 1$ 'zero's starting and ending at the state labelled 1 is just λ^{-g} .

4.4.1 Spectrum of maxentropic RLL sequences

If a transmitter emits the phrases T_j independently with probability $Pr(T_j)$, then the power spectral density function of the corresponding RLL sequence has been derived by Gallopoulos, Heegard & Siegel [110] and Pelchat & Geist [279, 280]

$$H(\omega) = \frac{1}{\bar{T} \sin^2 \omega/2} \frac{1 - |G(\omega)|^2}{|1 + G(\omega)|^2}, \quad (4.30)$$

where

$$G(\omega) = \sum_{l=d+1}^{k+1} Pr(T_l) e^{j\omega l} \quad (4.31)$$

and

$$\bar{T} = \sum_{l=d+1}^{k+1} l Pr(T_l). \quad (4.32)$$

An elegant proof, taken from Gallopoulos *et al.* [110], using generating functions, is given in the Appendix of this chapter.

The runlengths of a maxentropic sequence follow, as earlier argued, a truncated geometric distribution with parameter λ , or

$$Pr(T_l) = \lambda^{-l}, \quad l = d + 1, d + 2, \dots, k + 1, \quad (4.33)$$

whence

$$\bar{T} = \sum_{l=d+1}^{k+1} l Pr(T_l) = \sum_{l=d+1}^{k+1} l \lambda^{-l}. \quad (4.34)$$

Substitution of the distribution provides a straightforward method of determining the spectrum (PSD) of maxentropic RLL sequences. Figure 4.5 depicts the spectra $H(\omega)$ of maxentropic (d) sequences for selected values of the minimum runlength d . We may observe the following characteristics: maxima occur at non-zero frequency, and the spectra exhibit a more pronounced peak with increasing d . The energy in the low-frequency range diminishes with decreasing minimum runlength d . We can establish the following relationship for the power density at zero frequency:

$$H(0) = \frac{1}{\bar{T}} \sum_{l=d+1}^{k+1} (l - \bar{T})^2 Pr(T_l). \quad (4.35)$$

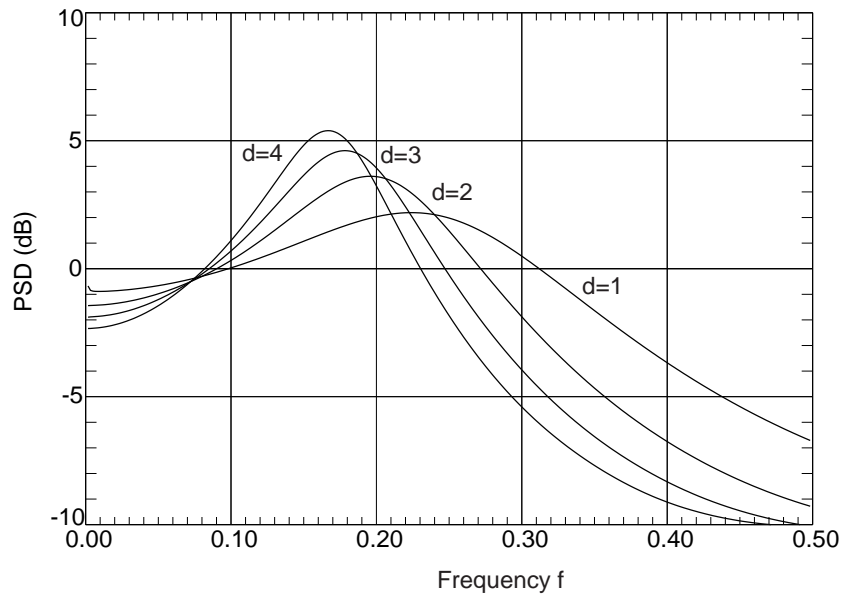


Figure 4.5: Power density function versus frequency of maxentropic runlength-limited sequences. Both the frequency scale and the pulse lengths of the RLL sequences are normalized in such a way that the user bit rate of all sequences is fixed at 1 bit/s. The vertical axis is scaled for unity total power in the bandwidth $0, \dots, 1/C(d, k)$.

Figure 4.6: Power density function versus frequency of maxentropic dk -constrained RLL sequences with $d = 2$ and selected k values.

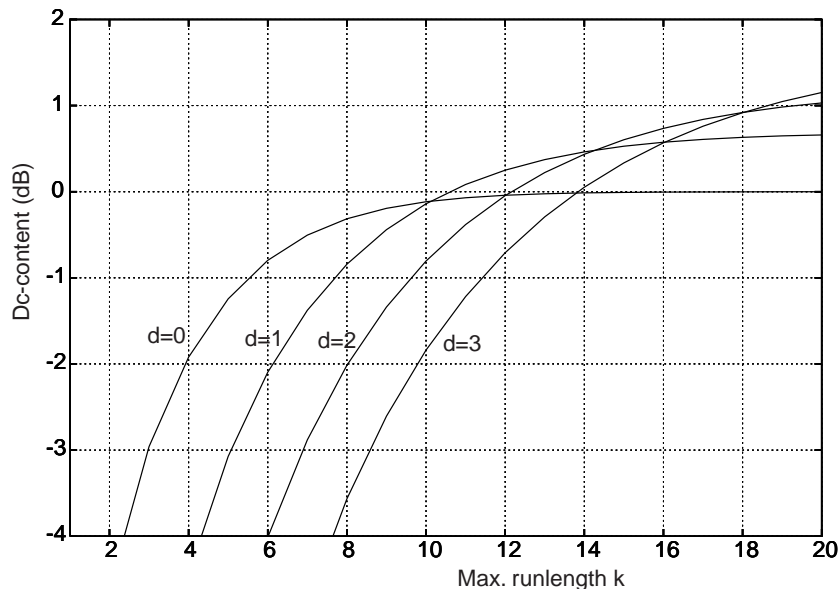


Figure 4.7: Dc-content of maxentropic dk -constrained RLL sequences versus the maximum runlength k with $d = 0, 1, 2, 3$. The relationship is plotted as a solid line. Note, however, that only a discrete set of points is achievable.

The effects on the spectra of the maximum runlength can be observed in Figure 4.6. The figure depicts the spectrum of maxentropic ($d = 2$) sequences with the maximum runlength k as a parameter. Figure 4.7 shows the power density at zero frequency (dc-content) of maxentropic dk -constrained RLL sequences versus k .

4.4.2 Comparison of results

In the next sections, a detailed description is offered on the design and implementation of codes. It will be shown that, for moderate hardware, encoders can be constructed that reach a code rate of up to 85–95% of the theoretical maximum. This fact motivated us to compute the runlength distribution and spectrum of two implemented codes and to compare them with the outcomes obtained by the preceding theory of maxentropic sequences. The implemented codes to be considered are the MFM code (see Chapter 5) with parameters $d = 1$ and $k = 3$ and the implemented $(2, 7)$ code (see later for more details).

The runlength distribution of the MFM code can be found by inspection of its state-transition diagram, see Example 3.2, page 42. The distribution of the implemented $(2, 7)$ code is taken from Howell [146]. Early spectral results by computer simulations have been presented by Shaft [297]. The runlength distributions of the two implemented codes are collected in Ta-

bles 4.6 and 4.7. The distributions of the corresponding maxentropic run-length constrained systems, discussed in the previous section, are included for comparison purposes.

Table 4.6: Runlength distribution of MFM code and its maxentropic counterpart.

T_i	$Pr(T_i)$ (<i>implemented</i>)	$Pr(T_i)$ (<i>maxentropic</i>)
2	0.500	0.466
3	0.333	0.318
4	0.167	0.217

Table 4.7: Runlength distribution of (2,7) code and its maxentropic counterpart.

T_i	$Pr(T_i)$ (<i>implemented</i>)	$Pr(T_i)$ (<i>maxentropic</i>)
3	0.336	0.341
4	0.257	0.238
5	0.181	0.166
6	0.128	0.116
7	0.076	0.081
8	0.023	0.057

The spectra of MFM and the implemented (2,7) code are plotted in Figures 4.8 and 4.9 along with their maxentropic counterparts. The spectrum of MFM is computed using the spectral analysis described in Example 3.2, and the expression of the spectrum of the (2,7) code is taken from Gallopoulos *et al.* [110]. We note a surprisingly good agreement (only a few dB difference) with the spectra of their maxentropic counterparts in the lower frequency range.

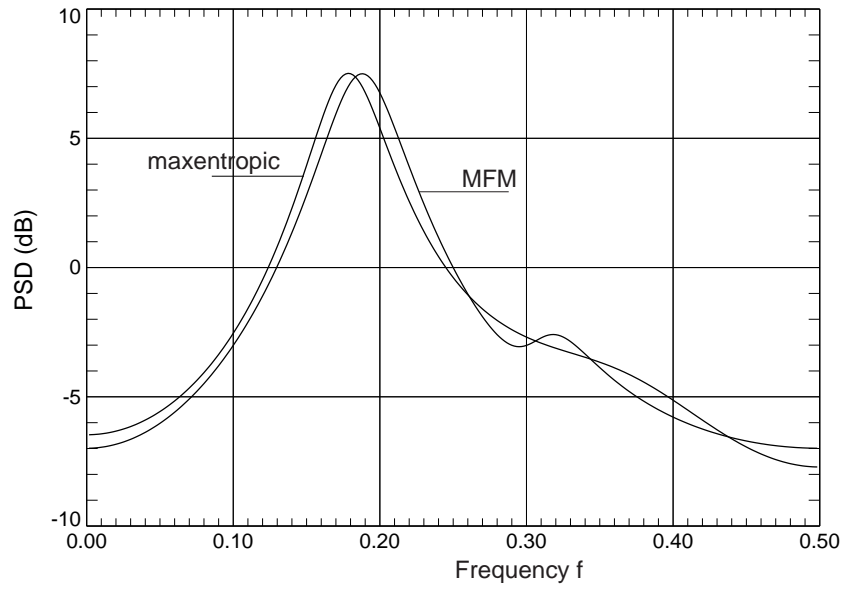


Figure 4.8: Spectrum of MFM code along with its maxentropic counterpart.

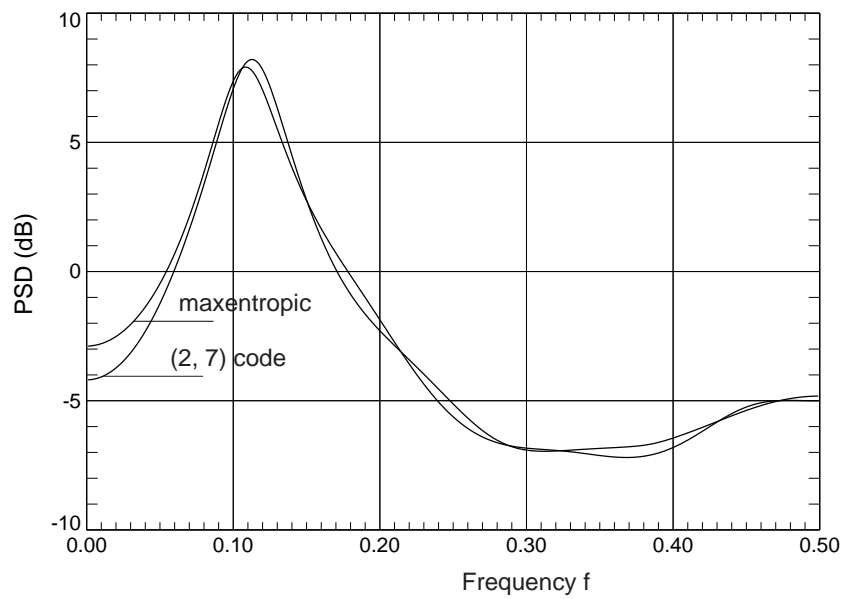


Figure 4.9: Spectrum of rate 1/2, (2,7) code along with its maxentropic counterpart.

4.5 Other runlength constraints

The theory of RLL sequences is not limited to (d, k) sequences. A variety of other runlength constraints have been developed. This section provides an overview of the most relevant examples.

4.5.1 Prefix-synchronized RLL sequences

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 specifically transmitted at the beginnings of the frames. Usually, the above arrangement is termed a *prefix-synchronized format* [162].

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 thus loses time to send 'useful' user data), 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.

In 1960, frame synchronization of unconstrained binary sequences in a general setting was addressed by Gilbert [111] (see also Stiffler [310], 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 Odlyzko [120] who provided elegant combinatorial arguments to establish closed form expressions for generating functions.

We commence, in Section 4.5.1, with a brief description of prefix synchronized 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.

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 denotes the set of integers $\{d + 1, \dots, k + 1\}$. The p -tuple $\mathbf{s} = (s_1, \dots, s_p)$ is used as a shorthand notation to describe the sync pattern. For example, $\mathbf{s} = (2, 1, 3)$ refers to the sync pattern '101100'. To keep notation to a minimum, we will freely abuse notation by referring to \mathbf{s} to represent both the p -tuple (s_1, \dots, s_p) and the string $10^{s_1-1} \dots 10^{s_p-1}$, which should be clear from the context. The length of the sync pattern, $L(\mathbf{s})$, is defined by

$$L(\mathbf{s}) = \sum_{i=1}^p s_i. \quad (4.36)$$

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 $L(\mathbf{s})$ or $L(\mathbf{s}) + 2d + 1$. The adopted definition (4.36) is slightly more convenient, as we will see shortly.

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

$$(T_{s_1}, T_{s_2}, \dots, T_{s_p}, T_{i_1}, T_{i_2}, \dots, T_{i_l}). \quad (4.37)$$

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

$$L_{\text{frame}} = \sum_{j=1}^p s_j + \sum_{j=1}^l i_j = L(\mathbf{s}) + \sum_{j=1}^l i_j. \quad (4.38)$$

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

$$(T_{s_2}, T_{s_3}, \dots, T_{s_p}, T_{i_1}, T_{i_2}, \dots, T_{i_l}, T_{s_1}, T_{s_2}, \dots, T_{s_{p-1}}) \quad (4.39)$$

agree with the sync pattern. Dictionaries satisfying this constraint are called *prefix-synchronized runlength-limited codes*. It is relevant to enumerate the number of distinct L_{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_{frame} is allowed to become very large.

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 [120] but can, in fact, be traced back to the work of Schuetzenberger on semaphore codes ([295], Remark 3), see also Berstel and Perrin, [28] II-7 and the notes following VI. Schuetzenberger attributes these results to Von Mises and Feller [81].

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

$$T = (T_{i_1}, \dots, T_{i_n}), \quad i_j \in dk, \quad j = 1, \dots, n, \quad (4.40)$$

composed of $n > p$ phrases of length

$$L(T) = \sum_{j=1}^n i_j, \quad (4.41)$$

with the properties that

$$(F1 :) \quad (i_1, \dots, i_p) = (i_{n-p+1}, \dots, i_n) = (s_1, \dots, s_p), \quad (4.42)$$

$$(F2 :) \quad (i_j, \dots, i_{j+p-1}) \neq (s_1, \dots, s_p), \quad j = 2, \dots, n-p. \quad (4.43)$$

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

$$f(z) = \sum_{T \in F} z^{-L(T)}. \quad (4.44)$$

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 . Note that the number $f_{L_{\text{frame}}+L(s)}$ is just the number of admissible frames. Following [120] we introduce two additional collections of sequences G and H , defined as follows. The collection G consists of all sequences T as in (4.40) composed of $n \geq p$ phrases such that

$$(G1 :) \quad (i_1, \dots, i_p) = (s_1, \dots, s_p), \quad (4.45)$$

$$(G2 :) \quad (i_j, \dots, i_{j+p-1}) \neq (s_1, \dots, s_p), \quad j = 2, \dots, n-p+1 \quad (4.46)$$

and H consist of all sequences T as in (4.40) composed of $n \geq 0$ phrases such that

$$(H1 :) \quad (i_j, \dots, i_{j+p-1}) \neq (s_1, \dots, s_p), \quad j = 1, \dots, n-p+1. \quad (4.47)$$

Note that by definition the empty sequence Λ is contained in H , $\mathbf{s} \in G$, 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

$$g(z) = \sum_{T \in G} z^{-L(T)} ; h(z) = \sum_{T \in H} z^{-L(T)}. \quad (4.48)$$

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 $G \setminus \{\mathbf{s}\}$, 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 $G \setminus \{\mathbf{s}\}$, 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

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

and sequences in

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

From this observation, the following lemma is immediate.

Lemma 4.5.1.1 $g(z)P_{dk}(z) = f(z) + g(z) - z^{-L(\mathbf{s})}$,

where

$$P_{dk}(z) = \sum_{i \in dk} z^{-L(T_i)} = \sum_{i \in dk} z^{-i}. \quad (4.49)$$

Proof: We have

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

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

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

and sequences in

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

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

Lemma 4.5.1.2 $P_{dk}(z)h(z) = h(z) + g(z) - 1$.

Proof: Similar to the proof of Lemma 4.5.1.1.

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 \mathbf{s} , respectively. Let $L(\mathbf{h}^{(i)})$ be the length of the first $p-i$ phrases of \mathbf{s} , or

$$L(\mathbf{h}^{(i)}) = \sum_{j=1}^{p-i} s_j. \quad (4.50)$$

The *auto-correlation function* of \mathbf{s} , denoted by \mathbf{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 \mathbf{r} is termed auto-correlation function. Let $\mathbf{s} = (1, 2, 1, 2, 1)$, then Figure 4.10 exemplifies the process of forming the auto-correlation function. If a marker tail coincides with a marker head, we have $r_i = 1$ else $r_i = 0$.

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

Figure 4.10: Process of forming the auto-correlation function \mathbf{r} .

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 4.5.1.3. 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

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

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 $U = \mathbf{s} * \hat{T} = (U_{i_1}, \dots, U_{i_{n+p}})$. Let j be the largest number such that $U_{i_j}, \dots, U_{i_{j+p-1}} = \mathbf{s}$. 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

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

From this observation, the following lemma easily follows.

Lemma 4.5.1.3 $(1 + F_r(z))g(z) = z^{-L(\mathbf{s})}h(z)$,

where the polynomial $F_r(z)$ is

$$F_r(z) = \sum_{i=1}^{p-1} r_i z^{L(\mathbf{h}^{(i)}) - L(\mathbf{s})}. \quad (4.51)$$

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

$$L(\mathbf{h}^{(i)}) - L(\mathbf{s}) = -L(s_1, \dots, s_i). \quad (4.52)$$

Note also that

$$r_0 z^{L(\mathbf{h}^{(0)}) - L(\mathbf{s})} = 1. \quad (4.53)$$

Therefore,

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

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

Theorem 4.2

$$z^{L(\mathbf{s})}f(z) = \frac{F_r(z)(P_{dk}(z) - 1) - z^{-L(\mathbf{s})}}{P_{dk}(z) - 1 - z^{-L(\mathbf{s})} - (1 - P_{dk}(z))F_r(z)}. \quad (4.54)$$

Proof: From Lemmas 4.5.1.2 and 4.5.1.3, an expression for $g(z)$ not involving $h(z)$ may be derived. If this expression for $g(z)$ is combined with Lemma 4.5.1.1, the theorem follows easily.

Corollary 4.5.1.1 *The number of admissible frames of length $N = L_{\text{frame}}$ is the coefficient of z^{-N} in the power series expression of the R.H.S. of (4.54).*

It is immediate from Theorem 4.2, that the number of constrained sequences is solely a function of the marker length $L(\mathbf{s})$ if the marker is repetitive free. Although Theorem 4.2 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.

Capacity

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

Theorem 4.3 $C(d, k, \mathbf{s}) = \log_2 \lambda$, where λ is the largest real root of

$$P_{dk}(z) - z^{-L(\mathbf{s})} - (1 - P_{dk}(z))F_r(z) = 1. \quad (4.55)$$

Proof: Follows from Theorem 4.2. Note that numerator and denominator of the R.H.S. of (4.54) have no common factors.

An upper and lower bound to the capacity $C(d, k, \mathbf{s})$ are given in the next Corollary.

Corollary 4.5.1.2 *For given sync pattern length $L(\mathbf{s})$, $\log_2 \lambda_l \leq C(d, k, \mathbf{s}) \leq \log_2 \lambda_u$, where λ_l is the largest real root of*

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

and λ_u is the largest real root of

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

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

Proof: Let

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

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

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

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

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

Table 4.8: Capacity of the (1,3)-constrained channel with prefix sync \mathbf{s} .

\mathbf{s}	$C(1,3,\mathbf{s})$
1010	.46496
10100	.45316
100010	.48673
100100	.50630
101010	.51737
1000100	.50902
1001010	.50902
1010010	.51606

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

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

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 λ_u is the largest real zero of $b(z)$, this implies $\lambda \leq \lambda_u$. Table 4.8 shows $C(1, 3, \mathbf{s})$ for selected sync patterns. From the table we observe, for example, that '100100' is the shortest sync pattern that admits of a code with rate 1:2. The shortest repetitive-free sync patterns that admit a code rate $> 1/2$ are '1000100' and '1001010'.

4.5.2 Asymmetrical runlength constraints

In optical data storage, one is often faced with an asymmetry between written and non-written data. With increasing information density the lengths of the non-written areas diminish, but the written effects (marks) cannot be made arbitrarily small. In practice, the minimum size of the recording marks is subject to limits. This depends on material parameters of the information layer and on the properties of the optical components by which the radiation beam is focused on the layer. Specifically in write-once and erasable optical recording, there is a significant asymmetry between marks and non-marks. Coding techniques based on asymmetrical RLL sequences may be quite successful in combating the effects of intersymbol interference [187, 69]. Other examples of asymmetrical runlength constraints, presented by Moon & Brickner, and Cideciyan *et al.* are maximum transition run (MTR) codes [58, 253, 254], where maximum runs of 'one's and 'zero's should be limited, see Section 4.5.3, page 81. Asymmetrical runlength-limited sequences are characterized by four parameters (d_0, k_0) and (d_1, k_1) , $d_0, d_1 \geq 0$ and $k_0 > d_0, k_1 > d_1$, which describe the constraints on alternate runlengths of 'zero's and 'one's, respectively. A permitted sequence is composed of alternate phrases of the form $0^i, i = d_0 + 1, d_0 + 2, \dots, k_0 + 1$, and $1^j, j = d_1 + 1, d_1 + 2, \dots, k_1 + 1$. We consider the time-multiplexing operation which forms a composite signal by repetitively interleaving phrases from two signal sets. Let one sequence be composed of phrases of durations $t_m \in S_o$, and let the second sequence have phrases of durations $t_j \in S_e$. The emitted, interleaved, sequence is composed of phrases taken alternately from the first, odd, sequence and the second, even, sequence. Reflection on the fact that the interleaved sequence is composed of phrases of duration $t_i = t_j + t_m, t_m \in S_o, t_j \in S_e$, will reveal that the characteristic equation is

$$\left(\sum_{j \in S_e} z^{-j} \right) \left(\sum_{m \in S_o} z^{-m} \right) = 1. \quad (4.56)$$

It is a conceptually simple matter to extend the preceding results to a larger number of multiplexed processes, and it is not further pursued here. From (4.56), we obtain the characteristic equation

$$\left(\sum_{i=d_0+1}^{k_0+1} z^{-i} \right) \left(\sum_{j=d_1+1}^{k_1+1} z^{-j} \right) = 1. \quad (4.57)$$

We will concentrate for a while on sequences with a minimum runlength constraint, i.e. $k_0 = k_1 = \infty$. Then (4.57) can be simplified in

$$z^{d_0+d_1+2} - 2z^{d_0+d_1+1} + z^{d_0+d_1} - 1 = 0. \quad (4.58)$$

As an immediate implication of the symmetry in d_0 and d_1 in the above expression, we find for the capacity of the asymmetrical RLL sequences

$$C_a(d_0, d_1, \infty, \infty) = C_a(d_0 + d_1, 0, \infty, \infty), \quad (4.59)$$

where $C_a(d_0, d_1, k_0, k_1)$ denotes the capacity of asymmetrical RLL sequences. Thus, the capacity of asymmetrical RLL sequences, $C_a(d_0, d_1, \infty, \infty)$, is a function of the sum of the two minimum runlength parameters only, and it suffices to compute $C_a(d_0, 0, \infty, \infty)$ by solving the characteristic equation

$$z^{d_0+2} - 2z^{d_0+1} + z^{d_0} - 1 = 0. \quad (4.60)$$

Results of computations are collected in Table 4.9.

Table 4.9: Capacity of asymmetrical RLL sequences versus minimum runlength.

d_0	$C_a(d_0, 0, \infty, \infty)$
1	0.8114
2	0.6942
3	0.6125
4	0.5515
5	0.5037

We can derive another useful relation with the following observation. Let $d_0 = d_1$, that is, the restrictions on the runlengths of 'zero's and 'one's are again symmetric, then from (4.59)

$$C_a(d_0, d_0, \infty, \infty) = C_a(2d_0, 0, \infty, \infty), \quad (4.61)$$

so that we obtain the following relation between the capacity of symmetrical, $C(d, k)$, and asymmetrical RLL sequences:

$$C_a(2d_0, 0, \infty, \infty) = C(d_0, \infty). \quad (4.62)$$

Katayama *et al.* [193, 194] presented a rate 8/15 asymmetrical $d_0 = 2, d_1 = 1$ runlength limited (RLL) code for dual-layered optical discs. The code features a minimum mark length of three channel bits to maintain the read-back signal power, and the minimum space length is two channel bits to maintain the partial-response maximum likelihood (PRML) performance. It has a better detection window size than EFMPlus, and the spectral density in the low-frequency band is lower than that of the EFMPlus code (see Chapter 11).

In the next section, we will discuss 'maximum transition run' (MTR) runlength constraints, which can be seen, for $d = 0$, as 'asymmetrical' runlength constraints.

4.5.3 MTR constraints

The Maximum Transition Run (MTR) codes, introduced by Moon & Brickner [58, 253, 254], have $d_0 = d_1 = 0$, and an asymmetrical constraint on the maximum runs of 0's and 1's. The maximum 'zero' runlength constraint, k_0 , is imposed, as in standard RLL constraints, for clock recovery, while the 'one' runlength constraint, k_1 , is imposed to bound the maximum number of consecutive transitions (i.e. consecutive 1's). It has been shown by Moon & Brickner [254] that removing those vexatious sequences leads to improved robustness against additive noise. The capacity of channels with an asymmetrical maximum runlength constraint, $C_a(0, 0, k_0, k_1)$, is easily found by invoking (4.57). The characteristic equation is

$$z^{k_0+1} + z^{k_1+1} + z^{k_0+k_1+4} - 2z^{k_0+k_1+3} = 1. \quad (4.63)$$

Results of computations are collected in Table 4.10 for the case $k_1 = 1$ and $k_0 = 1, \dots, 6$. Note that $C_a(0, 0, 1, 1)$ coincides with $C(0, 1)$ listed in Table 4.4, page 60. More results on asymmetrical constraints have been presented by Menyennett & Ferriera [245]. Moon & Brickner [254, 255] presented a rate 6/7, (0,0,1,5) [254], and a rate 16/17, (0,0,2,10) was designed by Nishiya *et al.* [262]. Van Wijngaarden & Soljanin presented a combinatorial technique for constructing high-rate MTR-RLL codes [348].

Table 4.10: Capacity of asymmetrical RLL sequences versus maximum 'zero' runlength k_0 . The maximum run of 'one's, k_1 , equals 1.

k_0	$C_a(0, 0, k_0, 1)$	k_0	$C_a(0, 0, k_0, 1)$
1	0.6942	4	0.8579
2	0.7947	5	0.8680
3	0.8376	6	0.8732

MTR (d, k) constraints, $d > 0$, have been advocated as they are said to improve the detection quality. The MTR constraint limits the number of consecutive strings of the form $0^d 1$, i.e. repetitive occurrence of the minimum runlength are limited. In wireless infrared communications applications, the MTR constraint is imposed as otherwise catastrophic receiver failure under near-field may be induced [123, 132]. Implementations of these codes usually have $d = 1$ and rate equal to 2/3. Rate 1/2, $d = 2$ codes have been presented by Lee.

4.5.4 RLL sequences with multiple spacings

Funk [105] showed that the theory of RLL sequences, as described in the previous sections, is unnecessarily narrow in scope. It precludes certain

relevant coding possibilities, which could prove useful in particular devices. The limitation is removed by introducing *multiple-spaced RLL sequences*, where one further degree of freedom is added, namely the runlength spacing, which is denoted by s . Let the parameters d and k again define the minimum and maximum allowable runlength. The runlength/spacing constraints may be expressed as follows: For integers d , k , and s , where $k - d$ is a multiple of s , the number of 'zero's between successive 'one's must be equal to $d + is$, where $0 \leq i \leq (k - d)/s$.

A sequence defined in this way is called an *RLL sequence with multiple spacings* (RLL/MS). Such a sequence is characterized by the 3-tuple (d, k, s) . Note that for standard RLL sequences, defined in Section 4.1, we have $s = 1$. Figure 4.11 illustrates a possible state-transition diagram.

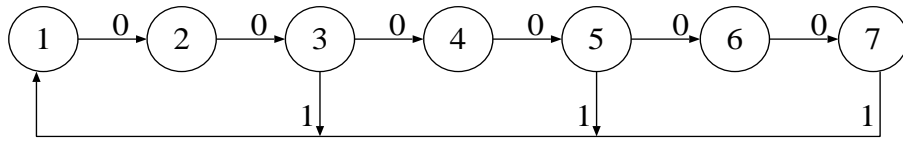


Figure 4.11: State-transition diagram for a $(d, k, s) = (2, 2, 6)$ sequence.

The capacity $C(d, k, s)$ can simply be found by invoking Shannon's capacity formula (2.27), page 25:

$$C(d, k, s) = \log \lambda, \quad (4.64)$$

where λ is the largest root of the characteristic equation

$$\sum_{i=0}^{(k-d)/s} z^{-(d+is+1)} = 1. \quad (4.65)$$

Note that if s and $d+1$ have a common factor, p , then $k+1$ is also divisible by p . Such a sequence is therefore equivalent to a $((d+1-p)/p, (k+1-p)/p, s/p)$ sequence. For $k = \infty$, we have

$$\sum_{i=0}^{\infty} z^{-(d+is+1)} = z^{-(d+1)} \sum_{i=0}^{\infty} z^{-is} = \frac{1}{1 - z^{-s}} z^{-(d+1)}. \quad (4.66)$$

Thus the characteristic equation is (see also the characteristic equation, (4.8), of regular (d) constrained sequences)

$$z^{d+1} - z^{d+1-s} - 1 = 0. \quad (4.67)$$

Table 4.11 shows results of computations. Within any s adjacent bit periods, there is only one possible location for the next 'one', given the location of the last 'one'. The detection window for an RLL/MS sequence is therefore

$T_w = sC(d, \infty, s)$, and the minimum time between two transitions, T_{\min} , equals $(d+1)C(d, \infty, s)$.

Table 4.11: Capacity $C(d, \infty, s)$ for selected values of d and s .

d	s	$C(d, \infty, s)$
0	2	0.6942
2	2	0.4057
3	2	0.3471
4	2	0.3063
0	3	0.5515
1	3	0.4057
2	3	0.3333

By rewriting (4.66), we obtain an interesting upper bound to T_w given a T_{\min} , which is similar to (4.15).

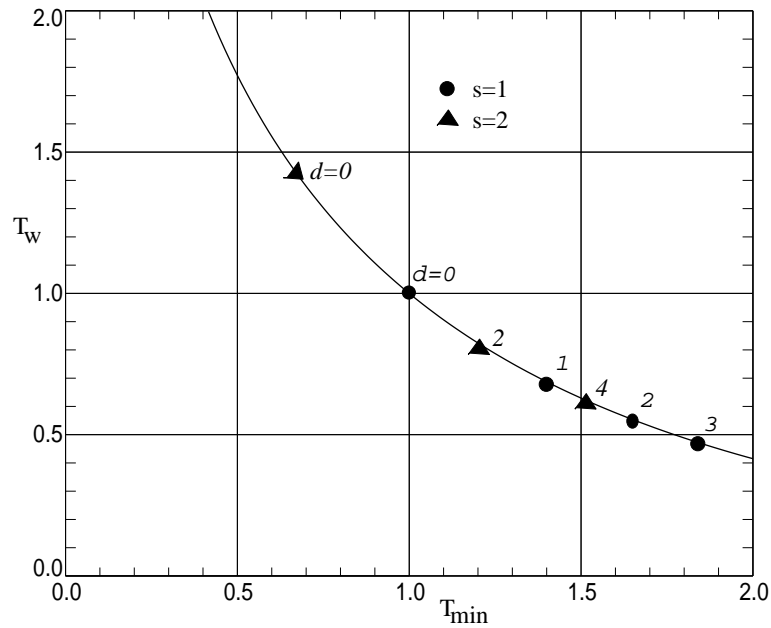


Figure 4.12: Relationship between T_{\min} and window T_w . The operating points of various (d, ∞, s) sequences are indicated.

Let

$$\lambda^{-s} + \lambda^{-(d+1)} = 1.$$

Then, by definition we have $\lambda = 2^{C(d, \infty, s)}$, so that

$$2^{-T_w} + 2^{-T_{\min}} = 1.$$

The above relationship is plotted in Figure 4.12. With (d) constrained sequences only discrete points on this curve are possible. RLL sequences with multiple spacing make it possible, by a proper choice of d and s , to achieve more points on this curve. The author does not know whether RLL/MS sequence have been used in practice.

4.5.5 $(O, G/I)$ sequences

Partial response signaling in conjunction with maximum likelihood detection [57, 207, 208, 351] is a data detection technique commonly used in magnetic recording. Special runlength constraints are needed to avoid vexatious sequences which could foil the detection circuitry. These constraints are characterized by two parameters G and I . The parameter G stipulates the maximum number of allowed 'zero's between consecutive 'one's, while the parameter I stipulates the maximum number of 'zero's between 'one's in both the even and odd numbered positions of the sequence. The G constraint, as the k constraint in dk sequences, is imposed to improve the timing. The I constraint is used to limit the hardware requirements of the detection circuitry. Marcus *et al.* [235] showed that it is possible to represent $(O, G/I)$ constraints by state-transition diagrams. To that end, we define three parameters. The quantity g denotes the number of 'zero's since the last 'one', and a and b denote the number of 'zero's since the last 'one' in the even and odd subsequence. It is immediate that

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

Each state in the state-transition diagram is labelled with 2-tuples (a, b) , where by definition $0 \leq a, b \leq I$ and $g(a, b) \leq G$. A transition between the states numbered by (a, b) to $(b, a + 1)$ (emitting a 'zero') and (a, b) to $(b, 0)$ (emitting a 'one') are easily attached.

Table 4.12: Capacity for selected values of G and I . Taken from [235].

G	I	capacity
4	4	0.9614
4	3	0.9395
3	6	0.9445
3	5	0.9415
3	4	0.9342
3	3	0.9157

By computing the maximum eigenvalue of the above state-transition matrix,

we obtain the capacity of the $(O, G/I)$ sequences. Results of computations are listed in Table 4.12.

Examples of implementation of $(O, G/I)$ constrained codes were given by Marcus, Siegel & Patel [232], Eggenberger & Patel [73] and Fitzpatrick & Knudson [84].

4.6 Weak constraints

Weakly constrained codes do not follow the letter of the law, as they produce sequences that violate the channel constraints with probability p . It is argued that if the channel is not free of errors, it is pointless to feed the channel with perfectly constrained sequences. In the case of a dk -constrained channel, violation of the d -constraint will very often lead to errors at the receiving site, but a violation of the k -constraint is usually harmless. Clearly, the extra freedom offered by weak constraints will result in an increase of the channel capacity, and possibly a higher code rate or a simpler code construction.

In the context of high-rate multi-mode codes, there is a growing interest in weakly constrained codes [155]. Multi-mode codes, as discussed in Chapter 10, operate by choosing the "best" word from a selection set of random words. Even if the selection size is enormous, the encoder can never fully guarantee the fulfillment of said channel constraints. There is always a non-zero probability that the selection set does not contain a proper codeword that satisfies the channel constraint.

4.6.1 Capacity of the weakly dk -constrained channel

In a weakly dk -constrained channel, (d, k) runlength conditions are disobeyed with probability p , i.e. with probability p a runlength of length i is transmitted that is either too short, $i < d$, or too long, $i > k$, [175]. The capacity of the weakly dk -constrained channel will be denoted by $C(d, k, p)$. Assume further that the probability a runlength i , $i > 0$, is transmitted equals p_i . By definition we have $\sum_{i \notin dk} p_i = p$ and $\sum_{i \in dk} p_i = 1 - p$, where dk denotes the set of integers $\{d + 1, \dots, k + 1\}$. Then, according to Section 2.3.2, page 24, the capacity of the weakly dk -constrained channel, $C(d, k, p)$, is by definition the maximum of the entropy function, $H()$, given by

$$H(d, k, p) = -\frac{\sum_{i=1}^{\infty} p_i \log_2 p_i}{\sum_{i=1}^{\infty} i p_i}, \quad 0 \leq p_i \leq 1, \quad (4.68)$$

under the conditions that

$$\sum_{i \in dk} p_i = 1 - p, \quad \sum_{i \notin dk} p_i = p.$$

Let

$$Q(z) = \sum_{i \in dk} z^{-i}$$

and

$$R(z) = \sum_{i \notin dk} z^{-i}.$$

According to [175] the capacity, $C(d, k, p)$, is given by the base-2 logarithm of the root of

$$(1 - p) \log_2 Q(z) + p \log_2 R(z) = p \log_2 p + (1 - p) \log_2(1 - p). \quad (4.69)$$

Results of computations for d -constrained channels are shown in Figure 4.13, which shows the capacity $C(d, \infty, p)$ of weakly d -constrained sequences as a function of the probability p that the sequences violate the d constraint. This diagram is meant to show a few essential characteristics of the capacity $C(d, \infty, p)$.

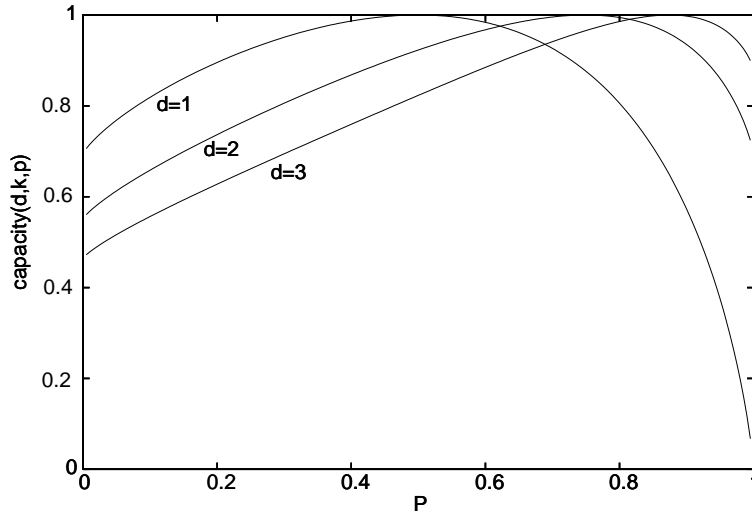


Figure 4.13: The capacity $C(d, \infty, p)$ of weakly d -constrained sequences as a function of the probability p that the sequences violate the d constraint.

We can observe, for example, that the capacity curves reach a maximum at $C(d, \infty, p') = 1$. At unity capacity the runlengths have the exponential distribution $p_i = 2^{-i}$, $i \geq 1$, and we therefore simply conclude that

$$p' = 1 - \sum_{i=d+1}^{k+1} 2^{-i}.$$

For d -constrained sequences we find

$$p' = 1 - 2^{-d}.$$

In addition we can establish the relationship

$$C(d, \infty, 1) = C(0, d - 1, 0), \quad d \geq 2.$$

Figure 4.14 deals with a more practical range of the violation probability p . The diagram shows the relative capacity gain $C(0, k, p)/C(0, k) - 1$ of k -constrained sequences. For the range of interest we find that the relative redundancy reduction is of the same order of magnitude as p .

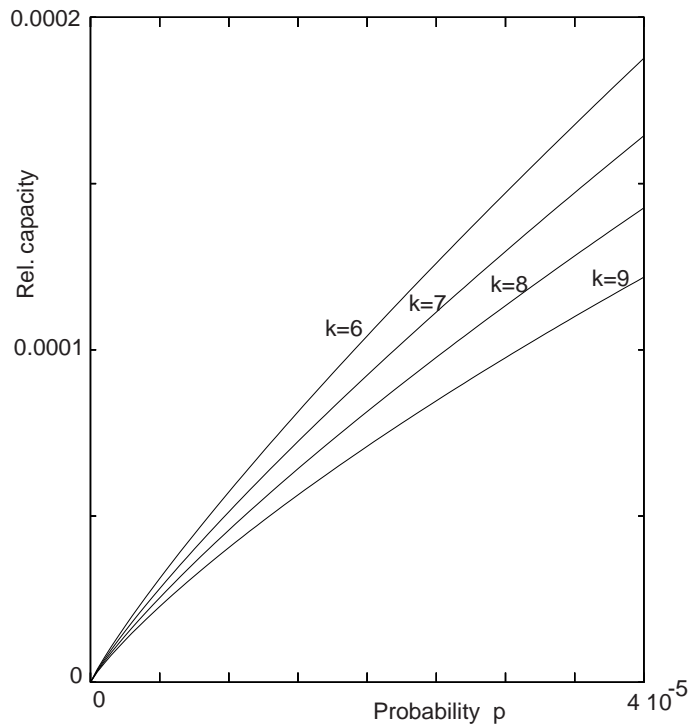


Figure 4.14: The relative capacity gain $C(0, k, p)/C(0, k) - 1$ of weakly k -constrained sequences as a function of the probability p that the sequences violate the k -constraint.

4.7 Multi-level RLL Sequences

Conventional recording channels only accept binary symbols, and this book follows the recording practice as it focuses on these specific sequences. There are, however, a few recording products, where more than two levels are used, so-called *multi-level* recording, and in this section we concentrate on multi-level RLL or dk -constrained sequences. Assuming an M -level or M -ary symbol alphabet $A = \{0, 1, \dots, M - 1\}$, an M -ary dk -constrained, or

$(d, k; M)$ -constrained, sequence is one with at least d and at most k 'zero's between nonzero symbols. Binary dk -constrained sequences are M -ary dk -constrained sequence with $M = 2$. Similarly, M -ary (d) sequences are $(d, \infty; M)$ -constrained sequences.

As in the case of binary dk -constrained sequences, $(d, k; M)$ -constrained sequences can be thought to be generated by a $(k + 1)$ -state finite-state machine. The adjacency, or connection, matrix, which gives the number of paths (words or symbols) going from state σ_i to state σ_j , is given by the $(k + 1) \times (k + 1)$ array D with entries d_{ij} , where

$$\begin{aligned} d_{i1} &= M - 1, \quad i \geq d + 1, \\ d_{ij} &= 1, \quad j = i + 1, \\ d_{ij} &= 0, \quad \text{otherwise.} \end{aligned} \tag{4.70}$$

A $(d, \infty; M)$ -constrained channel can be modelled by a finite-state machine of $d + 1$ states. The connection matrix is given by the $(d + 1) \times (d + 1)$ array D with entries d_{ij} , where

$$\begin{aligned} d_{ij} &= 1, \quad j = i + 1, \\ d_{d+1,1} &= M - 1, \\ d_{1,1} &= 1, \\ d_{ij} &= 0, \quad \text{otherwise.} \end{aligned} \tag{4.71}$$

The capacity of M -ary dk -constrained, denoted by $C(d, k; M)$, can now be found. Applying (2.19), page 22, we find that the capacity of an M -ary Markov source that emits dk -constrained sequences is

$$C(d, k; M) = \log_2 \lambda,$$

where λ is the largest real root of the characteristic equation

$$\det[D - zI] = 0,$$

and I is the identity matrix.

We will now do some counting of M -ary (d) sequences. Let $N_d(n; M)$ denote the number of distinct M -ary (d) sequences of length n , and define

$$\begin{aligned} N_d(n; M) &= 0, \quad n < 0, \\ N_d(0; M) &= 1. \end{aligned} \tag{4.72}$$

The number of M -ary (d) sequences of length $n > 0$ is found with the recursive relations [319]

$$\begin{aligned} N_d(n; M) &= n(M - 1) + 1, \quad 1 \leq n \leq d + 1, \\ N_d(n; M) &= N_d(n - 1; M) + (M - 1)N_d(n - d - 1; M), \quad n > d + 1. \end{aligned} \tag{4.73}$$

The characteristic equation of M -ary (d) sequences is

$$z^{d+1} - z^d - (M - 1) = 0. \quad (4.74)$$

In a similar fashion, the characteristic equation of M -ary (dk) sequences is

$$z^{k+2} - z^{k+1} - (M - 1)z^{k-d+1} + M - 1 = 0. \quad (4.75)$$

McLaughlin *et al.* [242] (see also Theorem 4.1, page 62, for the binary, $M = 2$, case) showed that the capacity of M -ary (dk) sequences is irrational for all values of M , d , and $k < \infty$. The capacity of M -ary (d) sequences is rational when

$$M = 1 + 2^{md}(2^m - 1)$$

for all values of the integer m larger than 0. In the following example, we will show some results for the case $d = 1, k = \infty$.

Example 4.2 Let $d = 1$, then $N_1(0; M) = 1$ and $N_1(1; M) = M$. $N_1(n; M)$, $n > 1$, is given by the simple first-order recursion equation

$$N_1(n; M) = N_1(n - 1; M) + (M - 1)N_1(n - 2; M), \quad n > 1.$$

Then $N_1(n; M)$ is given by

$$N_1(n; M) = a_1\lambda_1^n + a_2\lambda_2^n, \quad n \geq 0,$$

where

$$\lambda_{1,2} = \frac{1 \mp \sqrt{1 + 4(M - 1)}}{2},$$

and the two constants a_1 and a_2 can be found from $N_1(0; M) = 1 = a_1 + a_2$, and $N_1(1; M) = M = a_1\lambda_1 + a_2\lambda_2$. Let $d = 1$ and $M = 1 + 2^m(2^m - 1)$, $m > 0$, then the capacity is rational and equals m [242]. The number of constrained M -ary ($d = 1$) sequences of length n is given by

$$N_1(n; M) = a_1 2^{mn} + a_2 (1 - 2^m)^n, \quad n \geq 0,$$

where $a_2 = (M - 2^m)/(1 - 2^{m+1})$ and $a_1 = 1 - a_2$. For example, for $m = 1$, we have $M = 3$, and we simply obtain

$$N_1(n; 3) = \frac{1}{3}(2^{n+2} + (-1)^{n+1}) = \text{round}(2^{n+2}/3), \quad n \geq 0.$$

We can easily check that $C(1, \infty; 3) = 1$.

Note that in case the capacity is rational that 100% efficient codes are possible in theory. The remarkable thing is that indeed such implementations of 100% efficient encoders can be found. A 2-state encoder, whose encoder table is given in Table 4.13, of unity rate can achieve 100% efficiency for the 3-level ($d = 1$)-constrained channel.

Table 4.13: Codebook of two-state $R = 1$, $d = 1$, $M = 3$ code.

β	$h, g(\sigma_1, \beta)$	$h, g(\sigma_2, \beta)$
0	0, σ_1	1, σ_1
1	0, σ_2	2, σ_1

The encoded sequence can be decoded by observing two consecutive channel symbols. McLaughlin [240] and Datta & McLaughlin [67, 68] have given other examples of M -ary RLL codes.

4.8 Two-dimensional RLL constraints

In conventional recording systems, information is organized along tracks. Interaction between neighboring tracks during writing and reading of the information cannot be neglected. During reading, in particular when tracking, either dynamic or static, is not optimal, both the information track itself plus part of the neighboring tracks are read, and a noisy phenomenon, called *crosstalk*, or *inter-track interference* (ITI) may disturb the reading process. Crosstalk is usually modelled as additive noise, and thus, essentially, the recording process is considered to be one-dimensional. Advanced coding systems that take into account inter-track interference, were developed by Soljanin & Georghiades [306].

It is expected that future mass data systems will show more of their two-dimensional character: the track pitch will become smaller and smaller relative to the reading-head dimensions, and, as a result, the recording process has to be modelled as a two-dimensional process. An example of a type of code, where the two-dimensional character of the medium is exploited to increase the code rate was introduced by Marcellin & Weber [230]. They introduced *multi-track* (d, k) -constrained binary codes. Such n -track codes are extensions of regular (d, k) codes for use in multi-track systems. In an n -track (d, k) -constrained binary code, the d constraint is required to be satisfied on each track, but the k constraint is required to be satisfied only by the bit-wise logical "or" of n consecutive tracks. For example, assume two parallel tracks, where the following sequences might be produced by a 2-track (d, k) code:

```
track 1  000010100010100
track 2  010000010000001.
```

Note that the $d = 1$ constraint is satisfied in each track, but that the $k = 2$ constraint is satisfied only in a joint manner –there are never more than two consecutive occurrences of '0' on both tracks simultaneously. Although

n -track codes can provide significant capacity increase over regular (d, k) codes, they suffer from the fact that a single faulty track (as caused by media defects, for example) may cause loss of synchronization and hence loss of the data on all tracks. To overcome this flaw Swanson & Wolf [311] introduced a class of codes, where a first track satisfies the regular (d, k) constraint, while the k -constraint of the second track is satisfied in the "joint" manner. Orcutt & Marcellin [267, 268] computed the capacity of *redundant* multi-track (d, k) -constrained binary codes, which allow only r tracks to be faulty at every time instant. Vasic computed capacity bounds and spectral properties [327, 330, 328]. Further improvements of n -track systems with faulty tracks were given by Kj & Marcellin [198].

In holographic recording, data is stored using optical means in the form of two-dimensional binary patterns. In order to safeguard the reliability of these patterns, certain channel constraints have been proposed. More information on holographic memories and channel constraints can be found in [126, 127].

Codes that take into account the two-dimensional character have been investigated by several authors. Talyansky, Etzion & Roth [315] studied efficient coding algorithms for two types of constraints on two-dimensional binary arrays. The first constraint considered is that of the t -conservative arrays, where each row and column of the array has at least t transitions. The second constraint is that of two-dimensional DC-free arrays, where in each row and each column the numbers of 1's equal the number of 0's. Blaum, Siegel, Sincerbox & Vardy [35, 36, 326] disclosed a code which eliminates long periodic stretches of contiguous light or dark regions in any of the dimensions of the holographic medium such that interference between adjacent images recorded in the same volume is effectively minimized.

Kato & Zeger [195] considered two-dimensional RLL constraints. A two-dimensional binary pattern of 1's and 0's arranged in an $m \times n$ rectangle is said to satisfy a two-dimensional (d, k) constraint if it satisfies a one-dimensional (d, k) -constraint both horizontally and vertically. The (d, k) -capacity is defined as

$$C(d, k) = \lim_{n, m \rightarrow \infty} \frac{1}{mn} \log_2 N_{d, k}(m, n), \quad (4.76)$$

where $N_{d, k}(m, n)$ denotes the number of valid $m \times n$ rectangles. In contrast to the one-dimensional capacity, as discussed in the main part of this chapter, there is little known about the two-dimensional capacity. It was shown by Calkin & Wilf that $C(d, k)$ is bounded as $0.587891 \leq C(d, k) \leq 0.588339$ [46]. Bounds on $C(d, k)$ have been derived by Kato & Zeger [195] and Siegel & Wolf [303]. They showed, among others, that $C(d, k) = 0$ if and only if $k = d + 1$, $d \geq 1$, $k > d$. Etzion [75] considered the coding question of how to cascade two arrays with the same runlength constraints horizontally and

vertically, in such a way that the runlength constraints will not be violated at the array boundaries.

4.9 Appendix: Computation of the spectrum

Let $\{y_i\}$, $y_i \in \{-1, 1\}$ be a bipolar RLL sequence. For reasons of mathematical convenience we define the sequence $\{x_i\}$ by

$$x_i = \frac{1}{2}(y_i - y_{i-1}),$$

where, clearly $x_i \in \{-1, 0, +1\}$. It should be appreciated that $\{x_i\}$ is not a (dk) sequence, a (dk) sequence can be obtained by the process $|x_i|$. Let $H_x(\omega)$ and $H_y(\omega)$ denote the power spectral density functions of $\{x_i\}$ and $\{y_i\}$, respectively, then we obtain the relationship

$$H_y(\omega) = \frac{1}{\sin^2 \omega/2} H_x(\omega). \quad (4.77)$$

We apply the z -transform to obtain the spectrum of $\{x_i\}$:

$$H_x(z) = \sum_{m=-\infty}^{\infty} R_x(m) z^m, \quad (4.78)$$

where $R_x(m) = E\{x_j x_{j+m}\}$ denotes the m th auto-correlation coefficient. By inspection, we find for the quantity

$$x_j x_{j+m} = \begin{cases} 0, & \text{if } x_j \text{ or } x_{j+m} = 0 \\ +1, & \text{if } x_j \neq 0, x_{j+m} \neq 0, \# \text{ phrases is even} \mid \text{run started} \\ -1, & \text{if } x_j \neq 0, x_{j+m} \neq 0, \# \text{ phrases is odd} \mid \text{run started.} \end{cases}$$

Thus

$$\begin{aligned} R_x(m) &= E\{x_j x_{j+m}\} \\ &= Pr(x_j \neq 0) \{Pr(x_{j+m} \neq 0, \text{ even}) - Pr(x_{j+m} \neq 0, \text{ odd})\}. \end{aligned}$$

Define

$$\Phi_l(z) = \sum_{j=1}^{\infty} z^j Pr(\text{Length of } l \text{ consecutive } T_i \text{ is } j \mid \text{run started}).$$

The coefficient of z^j corresponds to the probability associated with all sequences of l runs of total length j . It follows that

$$\begin{aligned} & \sum_{l=0,2,4,\dots} \Phi_l(z) - \sum_{l=1,3,\dots} \Phi_l(z) \\ &= 1 + \sum_{j=1}^{\infty} z^j \{Pr(\text{Length of even number of consecutive } T_i \text{ is } j \mid \text{run started}) \\ & \quad - Pr(\text{Length of odd number of consecutive } T_i \text{ is } j \mid \text{run started})\} \\ &= 1 + \sum_{j=1}^{\infty} z^j \{Pr(x_j = x_{j+m} \mid x_j \neq 0) - Pr(x_j = -x_{j+m} \mid x_j \neq 0)\} \\ &= 1 + \sum_{j=1}^{\infty} E\{x_j x_{j+m} \mid x_j \neq 0\} z^j. \end{aligned}$$

If the phrases T_j are randomly selected, we find

$$\Phi_l(z) = [\phi(z)]^l,$$

where

$$\phi(z) = \sum_i Pr(T_i)z^i.$$

From (4.78), we find

$$H_x(z) = \pi_0 \left\{ \sum_{l \text{ even}} [\Phi_l(z) + \Phi_l(z^{-1})] - \sum_{l \text{ odd}} [\Phi_l(z) + \Phi_l(z^{-1})] - 1 \right\},$$

where

$$\pi_0 = Pr\{x_i = 1\} = \frac{1}{\sum_j j Pr(T_j)}.$$

From the power series identity

$$\frac{1}{1-x^2} = \sum_{l=0}^{\infty} x^{2l}$$

we find

$$\begin{aligned} H_x(z) &= \pi_0 \left\{ \sum_{l=0,2,\dots} ([\phi(z)]^l + [\phi(z^{-1})]^l) - \sum_{l=1,3,\dots} ([\phi(z)]^l + [\phi(z^{-1})]^l) - 1 \right\} \\ &= \pi_0 \frac{1 - \phi(z)\phi(z^{-1})}{(1 + \phi(z))(1 + \phi(z^{-1}))}. \end{aligned}$$

After substitution of $z = e^{j\omega}$ and with a little rearrangement we obtain

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

where

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

and

$$\bar{T} = \frac{1}{\pi_0} = \sum_{l=d+1}^{k+1} l Pr(T_l).$$

Chapter 5

RLL Block Codes

5.1 Introduction

One approach that has proved very successful for the conversion of arbitrary source information into constrained sequences is the one constituted by block codes. The source sequence is partitioned into blocks of length m , called *source words*, and under the code rules such blocks are mapped onto words of n channel symbols, called *codewords*. In this chapter we will deal with block(-decodable) codes, where the observation of a single codeword suffices to retrieve the original source word. Evidently, block-decodable codes offer an advantageous solution relative to sliding-block codes since they make it easier to preserve a particular mapping between the source and the code symbols, and, obviously, error propagation is localized to one decoded m -block. Block-decodable codes are highly suitable in conjunction with Reed-Solomon error control codes. In the preferred embodiment of the coding system, the codewords have a 1-1 correspondence with the elements of the finite field $GF(2^m)$, thus enabling the construction of, for instance, a Reed-Solomon code directly over the dk -constrained codewords.

A block code may be state dependent encodable, in which case the codeword used to represent a given source block is a function of the channel or encoder state (say, the history), or the code may be state independent. State independence implies that codewords can be freely concatenated without violating the sequence constraints. A set of such codewords is called *self-concatenable*. The additional restriction of state-independency leads, in general, to codes that are more complex in terms of hardware than state-dependent codes for a given rate. State-independent decoding may be achieved for any well-designed (d, k) block code, even for state-dependent encoded codes, as will be indicated later.

We start with a description of (d, k) -constrained block codes, and present some examples to clarify the general setting. Thereafter, we define dk sequences with an additional constraint on the number of leading and trailing

'zero's. The additional constraints on the number of leading and trailing 'zero's will make it possible to easily construct block codes. Special attention is paid to high-rate $(0, k)$ block codes. In the final section of this chapter, codes are presented that are *almost-block-decodable* as decoding of the sequence can be accomplished by observing (part of) the received codeword plus a small part of the previous codeword. The Appendix describes, very briefly, so-called *generating functions*, an elegant method for computing the number of constrained sequences.

5.2 RLL (d, k) block codes

To clarify the concept of block-decodable codes, we have written down a simple illustrative case of a rate $3/5$, $(1, \infty)$ block code. It is assumed that the string of source bits, is partitioned into 3-bit source words. The codeword assignment of Table 5.1 provides a straightforward block code, which converts 3-bit source words into 5-bit codewords. The two left most columns tabulate the eight possible source words along with their decimal representation. We have enumerated all words of length four that comply with the $d = 1$ constraint. There are exactly eight such words (see also Table 4.2, page 55). The eight codewords, tabulated in the right hand column, are found by adding one leading 'zero' to the eight 4-bit words, so that, as a result, the codewords can be freely cascaded without violating the $d = 1$ constraint.

Table 5.1: Simple $(d = 1)$ block code.

	<i>source</i>	<i>output</i>
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 $m/n = 3/5 < C(1, \infty) \simeq 0.69$, where $C(1, \infty)$ denotes, see Chapter 4, the maximum rate possible for any code irrespective the complexity of the encoder. The *code efficiency*, designated by η , expressed as the quotient of code rate and capacity of the (d, k) -constrained channel

having the same runlength constraints, is

$$\eta = \frac{R}{C(d, k)} \simeq \frac{0.6}{0.69} \simeq 0.86. \quad (5.1)$$

Thus the simple look-up code considered above is sufficiently powerful to attain 86% of the rate that is maximally possible, which, actually, demonstrates that good efficiencies are feasible with very simple constructions. That this example is not untypical will be demonstrated in the remainder of this chapter. The decoding of the received codewords can in fact be effected 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 using the inverted assignment depicted in Table 5.1. In this example, 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 maximum runlength constraint can be incorporated in the code rules in a straightforward manner. For instance, in the ($d = 1$) code previously described, the first codeword symbol is preset to 'zero'. If, however, the last symbol of the preceding codeword and the second symbol of the actual codeword to be conveyed are both 'zero', then the first codeword symbol can be set to 'one' without violating the $d = 1$ channel constraint. This extra rule, which governs the selection of the first symbol, the *merging rule*, can be implemented quite smoothly with some extra hardware.

Table 5.2: Codebook of 2-state $R = 3/5$, (1,6) code.

i	$h, g(1, \beta_i)$	$h, g(2, \beta_i)$
0	00000, 2	10000, 2
1	00001, 1	10001, 1
2	00010, 2	10010, 2
3	00100, 2	10100, 2
4	00101, 1	10101, 1
5	01000, 2	01000, 2
6	01001, 1	01001, 1
7	01010, 2	01010, 2

It is readily conceded that with this additional 'merging' rule the (1, ∞) code, presented in Table 5.1, turns into a (1,6) code. The code efficiency

is now, as can be verified with Table 4.4: $\eta = 0.6/C(1,6) \simeq 0.6/0.669 \simeq 0.897$. The process of decoding is exactly the same as for the simple $(1, \infty)$ code, since the first bit, the 'merging' bit, is redundant, and in decoding it is skipped anyway. The $(1,6)$ code is a good illustration of a code that uses state-dependent encoding (the actual codeword transmitted depends on the previous codeword) and state-independent decoding (the source word can be retrieved by observing just a single codeword, that is, without knowledge of previous or upcoming codewords or the channel state). Table 5.2 shows the above rate $3/5$, $(1,6)$ code in the format of a finite-state machine encoder. The proper operation of the encoder can easily be verified.

It is straightforward to generalize the preceding implementation example to encoder constructions that generate sequences with an arbitrary value of the minimum runlength. To that end, choose some appropriate codeword length n . Write down all d -constrained words that start with d 'zero's. The number of codewords that meet the given runlength condition is $N_d(n-d)$, which can be computed with (4.2) or by using Table 4.2, page 55. A k constraint can be usually imposed by a more 'clever' use of the merging bits. The following subsection shows a worked example of a code called MFM.

5.2.1 Modified frequency modulation, MFM

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 disc drives. MFM is essentially a block code of length $n = 2$ with a simple merging rule in case the NRZI notation is employed. The MFM encoding table is shown in Table 5.3.

Table 5.3: Coding rules MFM code.

<i>Source</i>	<i>Output</i>
0	x0
1	01

The symbol indicated with 'x' is set to 'zero' if the preceding symbol is 'one' else it is set to 'one'. It can be verified that this construction yields a maximum runlength $k = 3$. MFM has high rate efficiency, $\eta \simeq 0.5/0.5515$, or approximately 91 %. Note that the encoding table does not follow the regular structure of a finite-state encoder. A graphical representation of the finite-state machine underlying the MFM code is pictured in Figure 5.1. The labelled edges emanating from a state define the encoding rule and the

state in which an edge terminates indicates the state, or coding rule to use next. 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'.

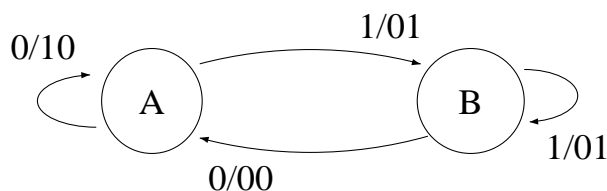


Figure 5.1: Two-state transition diagram that describes the MFM code in NRZI notation.

The finite-state encoder of MFM code discussed in Example 3.2, page 42, has four states. The explanation is that the change-of-state encoder used to translate a (dk) sequence into a runlength-limited (NRZ) sequence, accounts for one memory element, or a doubling of the number of states. Decoding of the MFM code is simply accomplished by discarding the redundant first bit in each received 2-bit codeword. In this section, cascading of sequences has been discussed in a rather heuristic fashion. The next section provides a more structured and general setting of this subject.

5.3 Block codes of minimum length

As was already pointed out, see Chapter 4, the dk constraints can be represented by a finite-state machines of $k + 1$ channel states. Note that in case in each state we have a sufficient number of outgoing codewords, that this machines can be simply used as a finite-state encoder by tagging input words to codewords. In this section, we discuss a method to modify this finite-state machines machine into a finite-state encoder by deleting a certain number of states. The crucial problem is to find a subset of the original channel states, referred to as *principal states*, which are used as the states of an encoder.

Let the size of the code be M , that is M source words can be accommodated. Then, from any principal state, there must emanate at least M words that terminate at the same or other principal states. The existence of such a set of principal states is therefore a necessary condition for the existence of a code with the specified number of source words. The condition implies that from any principal state, i.e. encoder state, there is a sufficient, $\geq M$, number of codewords that can be assigned to source words.

Franaszek [96] developed a recursive elimination technique for determining the existence of a set of principal states through operations on the state-transition matrix. The subsequent procedure, taken from Franaszek [95],

can be used to decide whether there exists a set of principal states for the specified parameters.

5.3.1 Franaszek's recursive elimination algorithm

Let the codeword length n and the source word length m be given. The size of the code is $M = 2^m$. 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 and $\sigma_i \in \Sigma^*$ a state to be tested. 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

$$\psi(\sigma_i, \Sigma^*) = \sum_{j \in V} [D]_{ij}^n, \quad V = \{j : \sigma_j \in \Sigma^*\}, \quad (5.2)$$

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

$$\psi(\sigma_i, \Sigma^*) \geq M,$$

thus $\Sigma^* = \Sigma_p$ is the set of principal states. The above 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 [220].

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

$$D^n \mathbf{v} \geq M \mathbf{v}, \quad (5.3)$$

where (in)equality means componentwise (in)equality. In our context, the vector \mathbf{v} is usually called the *approximate eigenvector*. It is not hard to see that the set $\{\sigma_{j_1}, \dots, \sigma_{j_L}\}$ for which $v_{j_1} = \dots = v_{j_L} = 1$ 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 5.1 We examine here the implementation of a rate 1/2, (1,3) code. Table 4.4 indicates that a code rate 1/2 represents 90% of the channel capacity. Therefore, let $m = 1$ and $n = 2$. The 2-step finite-state machine, FSM, graphs representing the (1,3) constraints are depicted in Figure 5.2.

Figure 5.2: Two-step FSM representing the (1,3) constraints.

Clearly, the state-transition matrix D^2 is

$$D^2 = \begin{bmatrix} 1 & 0 & 1 & 0 \\ 1 & 1 & 0 & 1 \\ 1 & 1 & 0 & 0 \\ 0 & 1 & 0 & 0 \end{bmatrix}. \quad (5.4)$$

Use of the recursive elimination algorithm indicates that state σ_4 has to be deleted (note that the row sum of row 4 is only one). We therefore eliminate row 4 and column 4. The 3×3 submatrix so obtained has row sums which are exactly 2. Thus, the principal states are σ_1 , σ_2 , and σ_3 . The codewords available for encoding associated with the principal states can be seen in Figure 5.3. States σ_2 and σ_3 are "equivalent" as they have edges with the same labels going to the same states, and they can be combined into a single state called state "2/3". The resulting diagram is shown in Figure 5.4. It is now, in the final step of the encoder construction, a matter of tagging the $2^m = 2$ source words to the outgoing edges of each state.

Figure 5.3: Same as Figure 5.2 with state σ_4 removed.

Figure 5.4: Two-state FSM after combining (merging) of states 2 and 3.

A two-state encoder, which is state-independently decodable, may be constructed with the assignments given in the following table:

i	$h, g(1, \beta_i)$	$h, g(2, \beta_i)$
0	00, 2	10, 2
1	01, 1	01, 1

After some rearrangement, we obtain the following simplified coding rules:

<i>source</i>	<i>output</i>
0	x0
1	01

A comparison with Table 5.3, page 98, or Figure 5.1 reveals that this is the MFM code.

Example 5.2 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 4.3, page 58)

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

It is plausible to suppose that fairly short codes exist with a rate $1/2$, and we proceed to show that such a block code indeed exists. Invoking the recursive elimination procedure previously described, one can prove that the shortest block code with $R = 1/2$ has a codeword length of 14. As

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

we conclude that all row sums ≥ 128 , so that all three states are principal states. To form a rate $7/14$ code, we are free to choose any 128 of the possible sequences

that leave a state and allocate them to the source words. In order to save in gate count we must follow a systematic approach. This particular example, however, is quite simple in this respect.

Perusal of the D^{14} matrix reveals that the lower-right element, $[D]_{3,3}^{14}$, is $129 > 128$. Thus only a single principal state, namely σ_3 , suffices for encoding. Apparently, this particular code can be state-independently encoded (all sequences start and end in state σ_3) and decoded. After a quick look at the state diagram in Figure 4.3, page 62, it will become apparent that any sequence terminating in σ_3 has at least $d = 2$ trailing 'zero's. In retrospect, this outcome is not surprising at all. When we take a look at Table 4.2, page 55, we notice there are exactly 129 ($d = 2$) sequences of length 12. As described in the previous section, the addition of two merging bits, which are preset to 'zero', completes the straightforward (alternative) design of a rate $7/(12+2)$, $(2, \infty)$ block code. Clearly, the 14-bit codewords can be freely cascaded without offending the prescribed runlength restrictions.

The shortest block codes of the specified rate for a selection of (d, k) combinations were computed by Franaszek [89] by invoking his recursive elimination procedure. Results of computations are collected in Table 5.4. The parameter $L = |\Sigma_p|$ denotes the minimum number of principal states.

Table 5.4: Shortest block codes of given bit-per-symbol values for a selection of dk constraints. After Franaszek [89].

d	k	m	n	L	$\eta = R/C(d, k)$
0	1	3	5	1	0.864
0	2	4	5	2	0.910
0	3	9	10	2	0.951
1	3	1	2	3	0.907
1	5	6	10	4	0.922
2	5	4	10	4	0.860
2	8	11	22	7	0.945
2	11	8	16	8	0.917
3	7	46	115	7	0.986
3	11	8	20	6	0.886

We also worked some examples of $d = 1$ and $d = 2$ codes. Tables 5.5 and 5.6 list the smallest codeword length possible for the specified rate and dk constraints with block-decodable codes. As can be seen from the tables, the codeword length required increases when the maximum runlength k is reduced, or in other words, when the actual rate of the code approaches the channel capacity for the specified conditions. We draw attention to the fact

that the minimum codeword lengths of the $R = 2/3$, (1,7) block code and the $R = 1/2$, (2,7) block code are quite large, namely 33 and 34, respectively. For these specific cases and many others, an alternative design, presented in Chapter 7, affords much shorter word lengths with much lower complexity and error propagation.

Table 5.5: Shortest block $R = 2/3$, ($d = 1$) code for a selection of k constraints.

k	n	$\eta = R/C(d, k)$
11	18	0.963
10	21	0.965
9	21	0.968
8	24	0.973
7	33	0.981
6	165	0.996

Table 5.6: Shortest block $R = 1/2$, ($d = 2$) code for a selection of k constraints.

k	n	$\eta = R/C(d, k)$
13	14	0.912
10	16	0.923
9	18	0.931
8	22	0.945
7	34	0.966

5.3.2 State-independent decoding

It can be verified that the MFM code dealt with in Example 5.1 may be uniquely decoded without knowledge of the encoder state. The capability of state-independent decoding is a notable virtue of a code. In some codes, the states may be functions of the instantaneous symbol sequence sum. Here a single error in detection may entail an unbounded string of decoding errors. Runlength constraints are less catastrophic since by observing at most k consecutive symbols of a dk -constrained sequence (d symbols are sufficient for a d -constrained sequence) we can remove the channel (or encoder) state ignorance at the receiver. Virtually all block codes possess the property of state-independent decodability.

The state-dependent translation of source words $\{\beta_u\}$ into codewords $\{\chi_{iu}\}$ with the output function $h(\sigma_i, \beta_u)$ must have an unambiguous inverse mapping $h^{-1}(\chi_{iu}) = \beta_u$ without reference to the encoder state σ_i . This concept is best illustrated by a codebook lacking this property. Consider a 3-state encoder and let $W(\sigma_i)$, $i = 1, 2, 3$, denote the sets of codewords emanating from the principal state σ_i . Let

$$\begin{aligned} W(\sigma_1) &= \{\chi_1, \chi_2, \chi_3, \chi_5\} \\ W(\sigma_2) &= \{\chi_1, \chi_3, \chi_4, \chi_5\} \\ W(\sigma_3) &= \{\chi_2, \chi_3, \chi_4, \chi_5\}, \end{aligned}$$

that is, four codewords leave each principal state. Table 5.7 shows a possible assignment of four source and codewords $h(\sigma_i, \beta_u)$. It can be verified that the given assignment does not admit state-independent decoding, and after some trial and error we may conclude that, in the given example, it is not possible to find any permutation of the source-channel representation that can be decoded without knowledge, at the decoder side, of the encoder state.

Table 5.7: Sets of codewords which do not admit state-independent decoding.

<i>source</i>	$h(\sigma_1, \beta_i)$	$h(\sigma_2, \beta_i)$	$h(\sigma_3, \beta_i)$
β_1	χ_1	χ_1	χ_2
β_2	χ_2	χ_4	χ_4
β_3	χ_3	χ_3	χ_3
β_4	χ_5	χ_5	χ_5

Since coder and decoder complexity increases exponentially with codeword length, it is usually advantageous to search for the shortest existing code without regard to state independence and try to achieve state-independent decodability by a proper allocation of source symbols to codewords. It is obvious that state-independent decoding is always possible when there are only two states. The next theorem, taken from Franaszek [89], asserts that state-independent decoding may be achieved for any (d, k) -constrained block code and any number of principal states.

Theorem 5.1 *Let $W(\sigma_i)$ denote the set of codewords of length n which take the symbol sequence from state σ_i through a succession of allowable states. Given a set of (principal) states $\hat{\Sigma} = \{\sigma_i\} \subset \Sigma$ and a class of associated word sets $W(\sigma_i)$ leaving σ_i such that each word set contains at least 2^m words, it is possible to allot source words β_u , $u = 1, \dots, 2^m$, to the word sets $W(\sigma_i)$ so that there is a unique inverse mapping.*

Proof: The state numbering follows the convention defined in Section 4.3.1, page 60. A word χ that is allowable from at least one principal state, i.e. which may be obtained from that principal state by an appropriate path through the state-transition diagram, is not allowable from another state either because there is a 'one' too close to the beginning of the word, or because there are too many 'zero's before the first 'one'. Suppose χ is permissible from state σ_v and σ_{v+u} , where $u > 0$, that is $\chi \in W(\sigma_v)$ and $\chi \in W(\sigma_{v+u})$. Then the allowability from σ_v implies that the first 'one' is not too close to the beginning for χ to be allowable from $\sigma_{v+1}, \sigma_{v+2}, \dots, \sigma_{v+u-1}$. The allowability of χ from σ_{v+u} implies that it does not have too many 'zero's in the prefix to be allowable from $\sigma_{v+1}, \sigma_{v+2}, \dots, \sigma_{v+u-1}$. Thus,

$$\chi \in W(\sigma_v) \cup W(\sigma_{v+u}) \Rightarrow \chi \in W(\sigma_{v+j}), \quad j = 1, 2, \dots, u-1.$$

Suppose that a source word β_e has been assigned to χ in the set $W(\sigma_v)$ then it is possible to give χ the same assignment in $W(\sigma_{v+1}), \dots, W(\sigma_{v+u})$, in other words, if $h(\sigma_v, \beta_e) = \chi$ then it is possible to allocate $h(\sigma_{v+1}, \beta_e), \dots, h(\sigma_{v+u}, \beta_e) = \chi$. Thus the inverse function is unique and independent of the encoder state, as was to be proved.

5.4 Block codes based on $(dklr)$ sequences

The construction technique discussed in the previous section has the drawback that it requires a (large) number of look-up tables for encoding. In principle, the number of required look-up tables equals the number of principal states, which, if the maximum runlength k is large, can be quite prohibitive. In Section 5.5.1, it will be shown that the number of look-up tables can be reduced to a value significantly less than k . Two look-up tables are required for popular values of d less than three. An alternative method, where only a single look-up table is needed was first described by Tang & Bahl in their pioneering paper [319]. They presented a method of inserting merging sequences of length $\beta = 2d$ between adjacent (dk) sequences so that the dk constraints are not violated. Evidently the method requires only one look-up table for generating the (dk) sequences. Some additional logic hardware is needed to determine the merging bits used to cascade the (dk) sequences. The following systematic construction technique, taken from Beenker & Immink [25], employing $(dklr)$ is a generalization of that idea. The advantage is that only $\beta = d$ merging bits are required. We start with the definition of $(dklr)$ sequences.

A $(dklr)$ sequence is a (dk) sequence with additional constraints on the number of leading and trailing 'zero's:

1. l constraint - the number of consecutive leading 'zero's of the sequence, that is, the number of 'zeros' preceding the first 'one', is at most l .

2. r constraint - the number of consecutive trailing 'zero's of the sequence, that is, the number of 'zero's succeeding the last 'one', is at most r .

The additional constraints on the number of leading and trailing 'zero's allow the design of more efficient block codes than is possible with the technique of Tang & Bahl [319].

The number of distinct $(dklr)$ sequences of length n , $n > k$, $k \geq 2d$, denoted by $N_{dklr}(n)$, can be found by adding the appropriate entries of the matrix D^n . To this end, assume that the finite-state machine, see Figure 4.2, page 61, occupies state σ_u , $1 \leq u \leq k+1$, at the start of an n -sequence. Then the number of starting 'zero's is at most $k+1-u$ and at least $\max\{0, d+1-u\}$. If the n -sequence terminates in σ_v , $1 \leq v \leq k+1$, then the n -sequence has $v-1$ trailing 'zero's. Thus $N_{dklr}(n)$, $l \leq k-d$, equals the number of distinct sequences that emanate from σ_{k+1-l} and terminate in one of the states $\sigma_1, \dots, \sigma_{r+1}$. Similarly, $N_{dklr}(n)$, $l > k-d$, equals the number of sequences that start in σ_{k+1-l} or $\sigma_{2k+2-l-d}$ and terminate in one of the states $\sigma_1, \dots, \sigma_{r+1}$. In summary,

$$N_{dklr}(n) = \begin{cases} \sum_{j=1}^{r+1} [D]_{k+1-l,j}^n, & l \leq k-d, n > k, \\ \sum_{j=1}^{r+1} [D]_{k+1-l,j}^n + [D]_{2k+2-l-d,j}^n, & l > k-d, n > k. \end{cases} \quad (5.5)$$

In the next section, an alternative method is given to compute the number of $(dklr)$ sequences using the theory of generating functions.

5.4.1 Generating functions

An elegant way to compute the number of $(dklr)$ sequences is provided by the theory of generating functions as discussed in Section 5.9. We first compute the generating function of the number of (dk) sequences that start and end with a 'one'. To that end, let

$$T(x) = \sum_{m=0}^{\infty} c_m x^m$$

denote the generating function of (dk) sequences that start and end with a 'one', i.e. the coefficient c_m equals the number of dk sequences of length m that start and end with a 'one'. As such a (dk) sequence can be seen to be built up of objects, in this context called phrases, of length $d+1$ to $k+1$, generating functions can be used to enumerate the number of sequences. For the (d, k) sequence we start with a 'one' and then choose phrases from the set

$$\{0^d 1, 0^{d+1} 1, \dots, 0^k 1\}.$$

We now get the generating function

$$T(x) = x(1 + P(x) + P(x)^2 + P(x)^3 + \dots),$$

where $P(x) = x^{d+1} + \dots + x^{k+1}$. By rewriting this expression we get

$$T(x) = \frac{q(x)}{p(x)} = \frac{x}{1 - (x^{d+1} + \dots + x^{k+1})} = \frac{x(1-x)}{1-x-x^{d+1}-x^{k+2}}. \quad (5.6)$$

Note that $p(1/x)$ is the characteristic equation (4.16). In order to compute the number of $(dklr)$ sequences, we must allow that the number of 'zero's at the beginning and end of the word lies between 0 and l and 0 and r , respectively. A $(dklr)$ sequence looks like

$$0^u 1 \dots 10^v,$$

where $0 \leq u \leq l$ and $0 \leq v \leq r$. The generating function of the middle part starting and ending with a 'one' equals $T(x)$. The 'zero's at the beginning and end of the sequence have generating functions

$$T_l(x) = 1 + x + \dots + x^l = \frac{1-x^{l+1}}{1-x}$$

and

$$T_r(x) = \frac{1-x^{r+1}}{1-x}.$$

After multiplying $T(x)$ with $T_l(x)$ and $T_r(x)$, respectively, we find the generating function of $(dklr)$ sequences, denoted by $T_{dklr}(x)$,

$$T_{dklr}(x) = \frac{x(1-x^{l+1})(1-x^{r+1})}{(1-x)(1-x-x^{d+1}-x^{k+2})}. \quad (5.7)$$

With an algebraic manipulation package it is now very easy to compute the number of $(dklr)$ sequences. The generating function of (d) sequences is

$$T_d(x) = \frac{1}{x^d(1-x-x^d)}.$$

5.4.2 Constructions 1 and 2

As already explained, $(dklr)$ sequences cannot, in general, be cascaded without violating the dk constraint at the codeword boundaries. Inserting a number, β , of merging bits between adjoining $(dklr)$ -sequences makes it possible to preserve the d and k constraints for the cascaded output sequence. Tang & Bahl [319] showed in their pioneering paper that (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 next describe two constructions of block codes with merging rules of mounting complexity and efficiency.

Construction 1: Choose d, k, r, l , and n' such that $r + l \leq k - d$ 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'. In other words, the code is *self-concatenable*. For symmetry reasons we may expect that the number of $(dklr)$ sequences is maximized if the 'zero's are equally distributed between the beginning and the end of the words, that is, if we choose $l = \lfloor (k - d)/2 \rfloor$ and $r = k - d - l$. A proof of this condition was given by Blake [34].

Example 5.3 The Group-Coded Recording (GCR) code, also known as '4/5 code', was 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 4.4, page 60, the capacity of a sequence with no runs of more than two 'zero's is $C(0, 2) \simeq 0.879$. Thus the efficiency of the GCR code is $\eta = 91\%$. From the $2^n = 32$ possible combinations of $n = 5$ bits, 15 are eliminated because of the $(d = 0, k = 2, l = 1, r = 1)$ constraints, leaving 17. One can be discarded to produce the 16 unique codewords required. The one left 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.

The rate-ineffectiveness of the d merging bits which are all set to 'zero' can be improved by the following construction provided the maximum runlength parameter k is not too small, i.e., $k \geq 2d$. Generalizations for small values of k have been presented by Weber & Abdel-Khaffar [338].

Construction 2: Choose d, k , and n' such that $k \geq 2d$ and $d \geq 1$. Let $r = l = k - d$ and $\beta = d$. Then the $(dklr)$ sequences can be cascaded without violating the specified dk constraints if the β merging bits are governed by the following rules, which can easily be implemented. Let an n' -sequence end with a run of s '0's ($s \leq r$) while the next sequence starts with t , $t \leq l$, leading '0's. Table 5.8 shows the merging rules for the $\beta = d$ merging bits. In order to demonstrate the efficiency of the codes based on Constructions 1 and 2 we will consider some specific cases. For $m = 8$ and $d = 1, \dots, 4$ and $k = 2d, \dots, 20$ we have selected $n = n' + \beta$ in such a way that the code rate R was maximized. Tables 5.9 and 5.10 give the results for $m = 8$ and $d = 1, \dots, 4$. 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.

Table 5.8: Merging rules of $(dklr)$ sequences.

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

Table 5.9: Block codes based on Construction 1, $m = 8$.

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

It may be noticed from the tables that on average there is a slight difference in the code efficiencies obtained by Constructions 1 and 2, approximately 3 % in favor of Construction 2. With Franaszek's recursive elimination technique it can be verified that the codes presented in Table 5.10 are of minimum length. Clearly, the longer the codewords, the closer the coding efficiency is to unity. In Section 6.3.3, page 147, we will analyze how close the rate of codes based on the above constructions can approach channel capacity, $C(d, k)$, when the codeword length n is allowed to be extremely large. That analysis shows that a codeword length n of approximately $n = 256$ suffices to approach capacity to within 0.1-0.5%. In the next section, we will discuss *optimal block codes*, which maximize the number of messages M (not necessarily a power of two) that can be transmitted.

Table 5.10: Block codes based on Construction 2, $m = 8$.

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

5.5 Optimal block codes

Franaszek's recursive elimination procedure, see Section 5.3.1, makes it possible to discern whether a (d, k) -constrained block code is possible for the given parameters n and code size $M = 2^m$. For certain applications it is of interest to establish the *optimal* block code, where the optimal code is the one that maximizes the number of messages (code size) M (not necessarily a power of two) that can be transmitted. By invoking the recursive elimination procedure it is straightforward to find the optimum M . Choose $M = 1$. Apply Franaszek's procedure. If such a code is possible, increase M by one and apply this procedure recursively until we reach an M' for which a code is not possible. Then $M = M' - 1$ is optimal and the set of principal states found for $M' - 1$ is called the optimal set of principal states.

Gu & Fuja [119] and Tjalkens [322] showed that such a recursive search technique is not required as we can immediately write down the optimal M and the optimal set of principal states. For $0 < d < k$, and word length $n \geq d$, the optimal M equals the number of (dk) sequences having at least d leading and at most $k - 1$ trailing '0's, i.e., M equals the number of $(d, k, k-d, k-1)$ sequences of length $n-d$. The optimal set of principal states is $\Sigma \setminus \{\sigma_{k+1}\}$. Chaichanavong & Marcus [56] have presented a more general approach including other constraints. The next subsection will detail Gu & Fuja's findings.

5.5.1 Set-concatenation

Gu & Fuja considered the following problem, which is of great relevance for block-decodable dk -constrained codes.

Let S_i be a set of disjoint codewords that obey the channel constraint, and let \mathcal{S} be a collection of disjoint sets S_i . Each source word, m_i , is represented by a member of the set of codewords S_i . A collection \mathcal{S} is said to be set-concatenable with respect to the constraint at hand if in any set S_i we can select (at least) one codeword that can be cascaded with any codeword in the collection \mathcal{S} of sets S_j without violating the prescribed constraint.

Example 5.4 Consider the $(d = 3, k = 6)$ code with codeword length $n = 10$ given by the collection $\mathcal{S} = \{S_0, \dots, S_8\}$, where

$$\begin{aligned} S_0 &= \{0000001000, 0100000010, 0010000001\}, \\ S_1 &= \{0000010000, 0100000100\}, \\ S_2 &= \{0000010001, 0100001000\}, \\ S_3 &= \{0000100000, 0100010000\}, \\ S_4 &= \{0000100001, 0100010001\}, \\ S_5 &= \{0000100010, 1000000100\}, \end{aligned}$$

$$\begin{aligned} S_6 &= \{0001000001, 1000001000\}, \\ S_7 &= \{0001000010, 1000010000\}, \\ S_8 &= \{0001000100, 1000010001\}. \end{aligned}$$

Perusal of the above example will verify that this collection has the set-concatenability property. By deleting one set we obtain eight sets, so that it is possible to construct a rate $3/10$, $(3,6)$ code.

Interesting questions are now: How do we construct such a collection \mathcal{S} , and how do we maximize the size of the collection. Essentially, the answers to these questions were given in the previous sections. Use Franaszek's recursive elimination procedure to find the set of principal states and construct a state-independently decodable code. The collection \mathcal{S} is easily obtained by reading the codewords from the codebook constructed. As such there is nothing new under the sun.

Gu & Fuja proved by construction that the maximum size of the collection is the number of dk -constrained words that start with at least d 0s and end with at most $k - 1$ 0s. They also showed that the size of the set S_i is at most

$$\left\lfloor \frac{k-2}{k-d+1} \right\rfloor + 2.$$

Therefore it is possible to implement an encoder with a look-up table consisting of at most $\lfloor (k-2)/(k-d+1) \rfloor + 2$ representations of the source word. It should be noted that the maximum size of any S_i does (in general) not equal the number of states required to implement the encoder by a formal finite-state encoder. Given such a finite-state encoder, we can, given the current encoder state, look up the correct representation from the set. Dedicated electronics can be designed for this purpose. For the cases of practical interest, where $d = 1$ and $d = 2$, we immediately conclude that at most two representations are required. The important cases, where $d = 1$ and $d = 2$ will be described in detail in the next two examples. For values of $d \geq 3$, similar collections can be constructed. They are not reproduced here, and the interested reader is referred to [119]. We start with an example where $d = 1$.

Example 5.5 Consider the case $d = 1$ and let \mathcal{C} be the set of $d = 1, k$ constrained codewords, $2 \leq k \leq \infty$, that start with at least $d = 1$ and end with at most $k - 1$ 'zero's. Construct for each $c \in \mathcal{C}$ a collection \mathcal{S} of sets as follows: $S_i = c$ if $c = \{01b\}$ and $S_i = \{10b, 00b\}$, where b represents the last $n - 2$ bits of the codeword $c \in \mathcal{C}$. As the number of words '00b' \leq the number of words '10b' we conclude that the size of the code is indeed equal to the number of dk -constrained words that start with at least $d = 1$ 'zero's and end with at most $k - 1$ 'zero's.

In the next example, details of $d = 2$ codes will be discussed.

Example 5.6 Consider the case $d = 2$ and let \mathcal{C} be the set of $d = 2, k$ constrained codewords, $2 \leq k \leq \infty$, that start with at least $d = 2$ and end with at most $k - 1$ 'zero's. Construct for each $c \in \mathcal{C}$ a collection \mathcal{S} of sets as follows: Let $S_i = \{00a, 01b\}$ or $S_i = \{00a, 10b\}$, where a and b represent the last $n - 2$ bits of the codeword $c \in \mathcal{C}$. As the number of words '00b' \leq the number of words '10b' plus the number of word '01b', we conclude that the size of the code equals the number of dk -constrained words that start with at least $d = 2$ 'zero's and end with at most $k - 1$ 'zero's.

It is of interest to compare Gu & Fuja's method with Construction 1 and 2. Construction 2, see Section 5.4.2, page 5.4.2, uses $(d, k, l = k - d, r = k - d)$ sequences of length $n - d$ that are merged with a simple merging rule. For $k \geq 2d$, Construction 2 is equivalent to constructing a set-concatenable collection. This construction, as can be verified, is optimal for $d = 1$, but is sub-optimal for $d > 1$. For instance, when we apply Construction 2 for the parameters $(d = 3, k = 6)$ and $n = 10$, we obtain the collection

$$\begin{aligned} S_0 &= \{0000001000, 1000001000, 0100001000, 0010001000\}, \\ S_1 &= \{0000010001, 0000100001, 0100010001\}, \\ S_3 &= \{0000100001, 1000100001\}, \\ S_4 &= \{0001000001\}, \\ S_5 &= \{0001000010\}, \\ S_6 &= \{0001000100\}. \end{aligned}$$

Clearly, this code has a lower rate than the one discussed in Example 5.4. The construction method of Gu & Fuja does not provide 'better' codes than Franaszek's recursive elimination method. The assignment of codewords that allows state-independent decoding, in the final part of Franaszek's construction yields more or less "automatically" a small number of channel representations. The great contribution to the art made by Gu & Fuja's method is the greater insight it provides over prior methods.

5.6 Examples of RLL (d, k) block codes

In this section we shall take a closer look at the various implementations of RLL (d, k) block codes. We start with a description of the *Eight to Fourteen Modulation (EFM)* and continue with a variety of construction for maximum runlength constrained channels. The codeword length of the examples provided are relatively small. Examples of k -constrained codes using (very) large blocks are presented in Section 6.3.4, page 150.

5.6.1 EFM

The designers of the Compact Disc system chose, after ample experimental evidence, a runlength-limited code with minimum runlength $d = 2$. They opted for a block structure. A source block length $m = 8$ is an adequate choice for the source words, since the entire source format (16-bit audio samples, etc.) is byte oriented. The construction of EFM rests entirely, with some additional tricks of the trade to be explained shortly, on Construction 2. According to Table 5.10, page 110, a rate $8/16$, $(2,10)$ block-decodable code can be immediately constructed with Construction 2. This code can accommodate 257 source words. It can be verified that $k < 10$ is not possible for a simple rate $8/16$, block-decodable code.

The selection of the two merging bits used to cascade the 14-bit sequences are governed by Table 5.8, page 110. There are instances where the two merging bits are not uniquely prescribed by the minimum and maximum runlength conditions, and we have freedom to transmit one of a number of possible merging words. This freedom of choice is utilized for minimizing the power density at the low-frequency end. As the two merging bits did not provide sufficient freedom for controlling the power density, it was decided to increase the number of merging bits to three, and as a result thereby reducing the rate from $8/16$ to $8/17$.

Chapter 11 provides an in-depth discussion of the low-frequency (lf) control of EFM and other codes. The same chapter also offers a comprehensive study of the relationship between channel capacity and lf-control. More details of EFM, such as coding tables, and so on, can be found in the patent literature [165].

5.6.2 Rate $8/9$, $(0, 3)$ code

According to Table 4.4, page 60, the capacity of a sequence with no runs of more than three 'zero's 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 'zero's. For the specified $(0, 3)$ constraints we find

$$D^9 = \begin{bmatrix} 208 & 108 & 56 & 29 \\ 193 & 100 & 52 & 27 \\ 164 & 85 & 44 & 23 \\ 108 & 56 & 29 & 15 \end{bmatrix}. \quad (5.8)$$

The existence of a block code can be ascertained with Franaszek's recursive elimination technique previously described. 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.

Alternatively, we can calculate that there are 293 ($d = 0, k = 3, l = 1, r = 2$) sequences that can be cascaded, as in Construction 1, without a merging rule.

This, in principle, concludes the discussion. Patel [275] 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 (see Chapter 6), 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, given by Patel (see Mee & Daniel [244], volume 2), affords simple and inexpensive encoder and decoder logic. The allocation is based on the 'divide and conquer' strategy. 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.

5.6.3 Other k -constrained codes

Both $(0, k)$ codes discussed in the previous section, are examples of rate $(n - 1)/n$ block codes. The rate of these codes are prohibitively low for current HDD recording products. A code rate of 16/17 or 24/25 is commonly found in modern products.

In this section, we will outline a family of simple high-rate k -constrained block codes, first presented by Imminck & Wijngaarden [168]. The code rate is $R = (n - 1)/n$, n odd, $n \geq 9$, and the maximum runlength is $k = 1 + \lfloor n/3 \rfloor$. The construction is particularly suited for schemes where encoding and decoding using table look-up is impractical, as in order to build the codeword, at most eight bits of the $(n - 1)$ -bit source word have to be altered. This has an immediate bearing on the size of the look-up tables and the worst-case error propagation.

Definitions

Let $k(\mathbf{x})$ be the maximum runlength of 'zero's in the word \mathbf{x} . Let $l(\mathbf{x})$ be the number of consecutive leading 'zero's of the word \mathbf{x} , that is, the number of 'zeroes' preceding the first 'one'. And let $r(\mathbf{x})$ be the number of consecutive trailing 'zeroes' of the word \mathbf{x} , that is, the number of 'zero's succeeding the last 'one'.

Preliminaries

Let the $(n - 1)$ -tuple $\mathbf{z} = (z_1, \dots, z_{n-1})$ be the binary source word and let $p = (n - 1) \div 2 + 1$.

Define the intermediate n -tuple $\mathbf{x} = (x_1, \dots, x_n)$ by $x_i = z_i$, $1 \leq i \leq p - 1$, $x_{i+1} = z_i$, $p \leq i \leq n$, and set the *pivot bit* $x_p := 1$. The left and right parts of the vector \mathbf{x} , $\mathbf{x}\mathbf{l}$ and $\mathbf{x}\mathbf{r}$, are defined by $x\mathbf{l}_i = z_i$, $x\mathbf{r}_i = z_{i+p}$, $1 \leq i \leq n \div 2$. A more judicious allocation of the source bits is possible in a byte-oriented system (see later). Set the pivot bit $x_p := 1$.

MAIN ALGORITHM, $k=1+\lfloor n/3 \rfloor$

Let $r = \lfloor k/2 \rfloor$; $l = k - r$, and define the intermediate n -tuple

$\mathbf{y} = (y_1, \dots, y_n)$, where $y_i = x_i$, $1 \leq i \leq n$

If $l(\mathbf{y}) \leq l$ and $r(\mathbf{y}) \leq r$ and $k(\mathbf{y}) \leq k$ then transmit \mathbf{y} as is.

If $(l(\mathbf{y}) \leq l$ and $k(\mathbf{x}\mathbf{l}) \leq k)$ and $(r(\mathbf{y}) > r$ or $k(\mathbf{x}\mathbf{r}) > k)$ then begin
 $y_{p-1} := 1; y_p := 0; y_{p+1} := 0; y_{n-r} := 1; y_{n-r+1} := x_{p-1}; y_{n-r+2} := x_{p+1};$
 transmit \mathbf{y} ; end;

If $(r(\mathbf{y}) \leq l$ and $k(\mathbf{x}\mathbf{r}) \leq k)$ and $(l(\mathbf{y}) > l$ or $k(\mathbf{x}\mathbf{l}) > k)$ then begin
 $y_{p-1} := 0; y_p := 0; y_{p+1} := 1; y_{l+1} := 1; y_{l-1} := x_{p+1}; y_l := x_{p-1};$
 transmit \mathbf{y} ; end;

If $(l(\mathbf{y}) > l$ or $k(\mathbf{x}\mathbf{l}) > k)$ and $(r(\mathbf{y}) > r$ or $k(\mathbf{x}\mathbf{r}) > k)$ then begin
 $y_{p-1} := 1; y_p := 0; y_{p+1} := 1; y_{l+1} := 1; y_{n-r} := 1; y_l := x_{p-1}; y_{n-r+1} := x_{p+1};$
 transmit \mathbf{y} ; end;

During decoding, the pivot symbol y_p is observed. If $y_p = 1$ then decoding is straightforward. If, on the other hand, $y_p = 0$ the bits y_{p-1} and y_{p+1} are used to uniquely re-constitute the original $(n - 1)$ -tuple. Note that in total at most eight bits of the original $(n - 1)$ -tuple \mathbf{z} are involved in the scheme, namely $y_1, y_2, y_3, y_{p-1}, y_{p+1}, y_{n-2}, y_{n-1}$, and y_n . In order to avoid error propagation in a byte-oriented system, these bits should be taken from one input byte. Then any error propagation resulting from an error in the pivot bit y_p occurred during transmission is confined to one decoded byte. In the next subsection, the algorithm is worked out for a rate 16/17, (0,6) code.

Description of a rate 16/17, (0,6) code

The code translates 16 bits (two bytes) of user data into 17 channel bits. The 17-bit codewords are characterized by the fact that they have at most six consecutive 'zero's and have at most three leading and at most three trailing 'zero's. Error propagation is limited as any single channel bit error made during retrieval will result in at most one decoded byte error. Let \mathbf{z}

$= (z_1, \dots, z_{16})$ be the 16-bit input word. The 17-bit word $\mathbf{y} = (y_1, \dots, y_{17})$ is obtained by shuffling:

z_1	z_9	z_{10}	z_{11}	z_2	z_3	z_4	z_{12}	1	z_{13}	z_5	z_6	z_7	z_{14}	z_{15}	z_{16}	z_8
-------	-------	----------	----------	-------	-------	-------	----------	---	----------	-------	-------	-------	----------	----------	----------	-------

Encoding algorithm

Define the Boolean variables (the '+' denotes the logical 'or'-function) $L1 = y_1 + y_2 + y_3 + y_4$, $L2 = y_2 + \dots + y_8$ and let $R1 = y_{14} + y_{15} + y_{16} + y_{17}$, $R2 = y_{10} + \dots + y_{16}$. Let further $L = L1L2$ and $R = R1R2$.

Transmission of a word is based on the following 4-step algorithm:

If LR then transmit \mathbf{y} as is.

If \overline{LR} then reshuffle and transmit:

z_1	z_9	z_{10}	z_{11}	z_2	z_3	z_4	1	0	0	z_5	z_6	z_7	1	z_{12}	z_{13}	z_8
-------	-------	----------	----------	-------	-------	-------	---	---	---	-------	-------	-------	---	----------	----------	-------

If $\overline{L\overline{R}}$ then reshuffle and transmit:

z_1	z_{12}	z_{13}	1	z_2	z_3	z_4	0	0	1	z_5	z_6	z_7	z_{14}	z_{15}	z_{16}	z_8
-------	----------	----------	---	-------	-------	-------	---	---	---	-------	-------	-------	----------	----------	----------	-------

If $\overline{\overline{LR}}$ then reshuffle and transmit:

z_1	z_{12}	0	1	z_2	z_3	z_4	1	0	1	z_5	z_6	z_7	1	0	z_{13}	z_8
-------	----------	---	---	-------	-------	-------	---	---	---	-------	-------	-------	---	---	----------	-------

Each modification made during the encoding process is uniquely identifiable and decoding can therefore be done in a straightforward fashion. Note that the source bits (z_1, \dots, z_8) , which constitute the 1st source byte, are transmitted unaltered. The retrieved source bits (z_9, \dots, z_{16}) , which constitute the 2nd source byte, are functions of the pivot bit y_5 . It can therefore easily be seen that error propagation due to any single channel bit error in the received codeword is restricted to at most one decoded byte error.

Variations of the above construction have been presented by Kuki & Saeki [209], Wijngaarden & Soljanin [349] and Aziz *et al.* [18, 19, 20, 21]. Their construction features a smaller maximum runlength at the cost of mounting error propagation.

5.6.4 Sequence replacement technique

The sequence replacement technique published by Wijngaarden *et al.* [345] is a recursive method for removing forbidden subsequences from a source word. The positions where the forbidden subsequences are removed, are encoded as binary words, and subsequently inserted at predefined positions of the codeword. The sequence replacement technique is attractive as the complexity of encoder and decoder is very low, and the method is very efficient in terms of rate-capacity quotient. Although the method can be used to remove various kinds of unwanted subsequences, we will demonstrate the principle of operation of the method by an application to $(0, k)$ codes, where sequences of runs of $k + 1$ consecutive 0s are removed.

Principle of operation

Wijngaarden *et al.* published three methods of which only one will be described below. Let $X = (x_1, \dots, x_{n-1})$ be an $(n - 1)$ -bit source word, which has to be translated into an n -bit code word $Y = (y_1, \dots, y_n)$. Obviously, the rate of the code is $(n - 1)/n$. The task of the encoder is to translate the source word into a $(0, k)$ -constrained word.

The encoder starts by appending a '1' to the $(n - 1)$ -bit source word, yielding the n -bit word, denoted by $X1$. The encoder scans (from right to left, i.e. from LSB to MSB) the word 'X1', and if this word does not have the forbidden subsequence 0^{k+1} , the vector $Y = X1$ is transmitted as is. If, on the other hand, the first occurrence of subsequence 0^{k+1} is found, we invoke the following procedure. Let the source word be denoted by $X_20^{k+1}X_11$, where, by assumption X_1 has no forbidden subsequence. The forbidden subsequence 0^{k+1} is removed yielding the $(n - k - 1)$ -bit sequence X_2X_11 . Let the forbidden subsequence start at position p_1 . The position p_1 is represented as a $(k + 1)$ -bit binary word, $1A_11$, where A_1 is the binary representation, in $k - 1$ bits, of the position p_1 . The tail '1' of X_2X_11 is replaced by $1A_110$, yielding $X_2X_11A_110$. Note that the sequence $X_2X_11A_110$ is of length n . If $X_2X_11A_110$ is free from other occurrences of the subsequence 0^{k+1} then $Y = X_2X_11A_110$. Otherwise, the encoder repeats the above sequence replacement procedure for the string X_2X_11 etc, until all forbidden subsequences have been removed. After s replacements, the codeword Y will be of the form

$$Z1A_s10 \dots A_210A_110,$$

where A_i are the binary representations of the positions p_i , $1 \leq i \leq s$, where starts of the strings 0^{k+1} occurred in the original sequence. The sequence Z is the remaining part of the source word, which is free from forbidden words. The decoder can uniquely undo the various replacements and shifts made by the encoder.

Example 5.7 We use the sequence replacement technique to translate the 20-bit word $X = '00000100000011000000'$ into a 21-bit codeword of a rate 20/21, (0,5) code. The two forbidden subsequences 0^6 occur at the positions $p_1 = 7$ and $p_2 = 15$. We, therefore, have $s = 2$, $A_1 = '0110'$ (=6), $A_2 = '0111'$ (=15), and $Z = '00000111'$. We obtain the codeword $Y = Z1A_110A_210$, which after substitutions yields $'000001111011010011110'$.

It has been shown by van Wijngaarden *et al.* that the codeword length n has to satisfy the following requirement:

$$n \leq 2^{k-1} + k + 1, \quad k \geq 2.$$

The redundancy of the sequence replacement code is

$$1 - R \approx 2 \cdot 2^{-k}, \quad k \gg 1.$$

From (4.3), page 60, we know that the redundancy of maxentropic k -constrained sequences is

$$1 - C(0, k) \approx \frac{1}{4 \ln 2} 2^{-k} = \frac{1}{2.77} 2^{-k}, \quad k \gg 1,$$

which reveals that the redundancy of the sequence replacement method is a factor of five away from optimal for $k \gg 1$.

5.6.5 High-rate (klr) codes with uncoded symbols

Section 5.6.3 deals with high-rate k -constrained codes that can be encoded and decoded with a very simple algorithm. An attractive feature of this family of codes is that irrespective of the codeword length n , $n > 16$, only eight of the n bits need to be encoded. The remaining $(n - 8)$ bits, called *uncoded* bits, are equal to the source bits. Following terminology of error correcting codes, the uncoded, or unconstrained, part of the codeword is called the *systematic part* of the codeword. The benefits of having a large systematic codeword part are twofold: error propagation is limited to the coded symbols, and hardware is limited to a look-up table of the coded part. Obviously, these benefits are very attractive, and it is therefore of practical interest to investigate codes with a large systematic codeword part. A straightforward way of achieving this is by *interleaving* coded and uncoded symbols. For example, constrained words generated by a rate IBM 8/9, (0,3) code (see Section 5.6.2) can be interleaved with uncoded symbols. An example of a rate 24/25, (0,11), based on the above IBM code, has been presented by Fisher & Fitzpatrick [82]. Another application of interleaving the IBM code for constructing a rate 16/17, (0,8) code has been disclosed by Sonntag [307]. Both McEwen *et al.* [239] and McClellan [237] found that

interleaving of a base $(0, k)$ code with uncoded 8-bit symbols may lead to very high-rate codes. Rates including 32/33, 48/49, 56/57, 72/73, 80/81, and others are used in magnetic recording (HDD) channels. Interleaving is a simple method, but, as can be seen it is far from optimal: a rate 16/17, $(0,6)$ code with the virtue of single byte error propagation can be constructed with the method outlined in the previous section. Patapoutian [272] *et al.* disclosed a rate 32/33, $(0,6)$ code with limited error propagation.

Interleaving is an obvious example of the generation of codewords that are formed of coded and uncoded symbols, but probably we can do better. The general question is: Can we find a set of (klr) n -bit codewords that can be uniquely translated into a set of source words and *vice versa* with a given, and preferably maximum, size of the systematic part of the codeword? The analysis given below, taken from van Wijngaarden & Immink [347], offers an elegant answer to this interesting and practical question. More results have been presented by Campello *et al.* [47]

We start with some definitions. Let \mathbf{X} denote the set of (klr) codewords, $\mathbf{x}_i = (\chi_{i,1}, \dots, \chi_{i,n})$, $0 \leq i \leq M - 1$, of length n , and let \mathbf{B} be the set of source words, $\boldsymbol{\beta}_i = (\beta_{i,1}, \dots, \beta_{i,m})$, $0 \leq i \leq 2^m - 1$, $M \geq 2^m$, of length m . Define the vector $\boldsymbol{\psi} = (\psi_1, \dots, \psi_n)$, whose elements are binary, i.e. 0 or 1. Let $\psi_i = 0$ if the symbol at position i is uncoded, and 1 if the symbol at position i is coded. That part of the codeword, where $\psi_i = 0$ is the systematic part of the codeword. The question is now to find a subset $\hat{\mathbf{X}}$ of \mathbf{X} , $|\hat{\mathbf{X}}| = 2^m$, such that a one-to-one mapping $f: \mathbf{B} \rightarrow \hat{\mathbf{X}}$ for all $\boldsymbol{\beta}_i$ is possible, where the function f satisfies the condition that if $\psi_j = 0$ we stipulate $\beta_{i,r_j} = \chi_{i,j}$, $r_j \in \{1, \dots, m\}$. It is irrelevant for the moment to which of the source symbols the uncoded symbols are assigned. Note, however, that in a larger system, where the channel code cooperates with an error correcting code, the assignment of the systematic and non-systematic part can be of great concern.

A little thought will make it clear that a given symbol position is systematic if all codewords have a 'zero' at that position. Clearly, a 'zero' at that position is the worst case that can happen, as a placement of a 'one' is always possible. A mapping is therefore possible if the number of (klr) sequences that have 'zero's at the symbol positions j , where $\psi_j = 0$, equals or exceeds 2^m . The number of such (klr) sequences can be computed with generating functions or by a manipulation of the adjacency matrix. In the sequel, we will demonstrate how this can be achieved by a manipulation of the adjacency matrix.

Let D be the $(k+1) \times (k+1)$ adjacency matrix that represents the $(0, k)$ constrained channel (see also Section 4.3.1, page 60). The entries of D are

given by

$$\begin{aligned} d_{i1} &= 1, \quad i \geq 1, \\ d_{ij} &= 1, \quad j = i + 1, \\ d_{ij} &= 0, \quad \text{otherwise.} \end{aligned} \tag{5.9}$$

Let D^0 be the $(k + 1) \times (k + 1)$ matrix, whose entries are defined by

$$\begin{aligned} d_{ij}^0 &= 1, \quad j = i + 1, \\ d_{ij}^0 &= 0, \quad \text{otherwise.} \end{aligned} \tag{5.10}$$

Note that the entries of D^0 equal those of D with the exception of the left column that pertain to emitted 'one's (see Figure 4.2, page 61). Then compute recursively, for $i = 1, \dots, n$

$$A_i = A_{i-1}\{(1 - \psi_i)D^0 + \psi_i D\},$$

where $A_0 = E$. Let $A = A_n$. Then the number of (klr) sequences, N_{klr}^ψ , with 'zero's at the positions, where $\psi_i = 0$ equals

$$N_{klr}^\psi = 2^{n-w(\boldsymbol{\psi})} \sum_{j=1}^{r+1} \psi_{n-j+1} [A]_{k+1-l, j}, \quad n > k.$$

The weight $w(\boldsymbol{\psi})$ denotes the number of 'one's in the vector $\boldsymbol{\psi}$. In order to find the maximum number of uncoded symbols for given k and n , we have to conduct an exhaustive search trying all possible vectors $\boldsymbol{\psi}$. Results of computations are listed in Table 5.11.

Table 5.11: Maximum size of the systematic part of rate $(n - 1)/n$, $(0, k, \lceil \frac{k}{2} \rceil, \lfloor \frac{k}{2} \rfloor)$ codes.

n	$k = 3$	$k = 4$	$k = 5$	$k = 6$	$k = 7$	$k = 8$	$k = 9$
8	1	4	4	4	7	7	7
9	1	4	5	5	5	8	8
17	-	4	6	9	9	12	13
25	-	2	7	12	13	15	17

The table shows that a rate 16/17, (0,6) code can be constructed with a systematic part of nine uncoded symbols. The vector $\boldsymbol{\psi}$ that maximizes the number of codewords is (00110001110001110), and the number of available codewords is $73728 > 2^{16}$. A look-up table (ROM) is required for the encoding of the remaining eight coded symbols. The simple construction given in Section 5.6.3 of a code with the same specifications requires eight instead of nine uncoded positions. The nine coded symbols can be encoded with some simple random logic. The rate 24/25, (0,9) code disclosed in Section 5.6.3 has 17 uncoded symbols, and can thus not be improved according to Table 5.11.

5.7 Block-decodable RLL Codes

In the previous (and in the upcoming) chapters, the terms (d, k) and RLL sequence have been used as synonyms, and the design of encoders that generate RLL sequences has been conducted by designing encoders that generate (d, k) sequences. The encoder is followed by a change-of-state encoder (precoder) that translates the (d, k) into RLL sequences prior to recording them on disk or tape. Sequences that are assumed to be recorded with such an intermediate change-of-state encoding (precoding) step are said to be given in *non-return-to-zero-inverse* (NRZI) notation, whereas sequences transmitted without such a step are referred to as *non-return-to-zero* (NRZ).

Until the early 90's, code designers believed there was no price to be paid for the above design procedure, and that there was no "difference" between codes in NRZ and NRZI representation. However, intuition turned out to be false. If we directly design an RLL code, that is, instead of the conventional design of a (d, k) code followed by a precoder, the RLL code may have better attributes. In particular, it is profitable in terms of error propagation to directly design RLL encoders. The RLL codes on which we will concentrate in the next section are block decodable, while at the same time they are simpler to implement than equivalent (d, k) block-decodable codes currently being used.

We start with a description of the basic idea of the RLL coding technique. In order to simplify the exposition, we confine ourselves for a while to sequences with a minimum runlength only (i.e. $k = \infty$). A generalization to sequences with both a minimum and a maximum runlength constraint is postponed to a later section. The following simple example will demonstrate the potential usefulness of the coding technique, and in fact it shows that, indeed, there is a significant difference between code designs based on NRZ or NRZI representations.

Consider the simple $(d = 1, \infty)$ RLL code described in Table 5.12. Note that the coding table is not presented in the regular format of a finite-state machine. The codeword length is four and the code can accommodate six source words, i.e., has size six. The left hand column shows the source symbols, represented by an integer between '0' and '5', while the right hand column shows the possible channel representations of the corresponding source word. Note that the RLL code is presented in NRZ notation, that is, sequences generated by this code should therefore not be followed by a change-of-state encoding operation (precoder). Four of the six source words, namely 0, 2, 4, and 5, are uniquely represented by a codeword. It is easily verified that these four codewords can be freely concatenated without offending the prescribed minimum runlength constraint, that is, in the cascaded sequence there are at least two consecutive 'one's or 'zero's.

Table 5.12: Codebook of a $d = 1$ RLL block code.

Source	Codewords
0	0000
1	0001/0110/1000
2	0011
3	0111/1001/1110
4	1100
5	1111

The remaining two source words, namely '1' and '3', are each represented by three alternative codewords. The source word '1' may be represented by '0001', '0110', or '1000', and the source word '3' may be represented by '0111', '1001', or '1110'. The encoder's choice depends on the last codeword transmitted and the next codeword to come. For this simple case, the reader can verify that by looking ahead and looking back at most *one* codeword in time, it is always possible to select (at least) one of the three alternative representations that fulfills the stated minimum runlength requirement. A casual glance at Table 5.12 is sufficient to convince the reader that the table has the virtue of a unique state-independent inverse, which means that observation of a single codeword is sufficient information for the retrieval of the corresponding source word. We conclude, therefore, that the code presented in Table 5.12 is a $(1, \infty)$ RLL block-decodable code. In contrast, a conventional dk -constrained block-decodable code with the same parameters constructed with Franaszek's method, Section 5.3.1, has size five, and we draw the fascinating conclusion that the above RLL block-decodable code has a slightly larger size.

The above table shows the code in a non-canonical format, but with little effort the above coding rules can be cast into the model of a standard finite-state encoder. The encoder graph has eight states, and the output and next-state functions can be represented by the following 8×8 matrix:

$$\begin{bmatrix} 0/0000 & 1/0000 & 2/0000 & 3/0000 & & 4/0000 & 5/0000 \\ 0/0110 & & 2/0110 & & 1/0001 & 3/0001 & 4/0000 & 5/0000 \\ 0/0011 & & 2/0011 & & 1/0011 & 3/0011 & 4/0011 & 5/0011 \\ 0/0111 & & 2/0111 & & 1/0111 & 3/0111 & 4/0111 & 5/0111 \\ 0/1000 & 1/1000 & 2/1000 & 3/1000 & & & 4/1000 & 5/1000 \\ 0/1110 & & 2/1110 & & 1/1001 & 3/1001 & 4/1001 & 5/1001 \\ 0/1100 & 1/1100 & 2/1100 & 3/1100 & & & 4/1100 & 5/1100 \\ 0/1111 & & 2/1111 & & 1/1111 & 3/1111 & 4/1111 & 5/1111 \end{bmatrix}.$$

The matrix elements are expressed in the form x/y , where x is the source word and y is the corresponding encoder output. For reasons of space, we follow the convention that if a transition is allowed from state i to state j ,

the i, j matrix element equals the label x/y pertaining to the edge $i \rightarrow j$. If, on the other hand, such a transition is not permitted, the matrix element is empty. The look-ahead dependence of the codewords was accounted for by tagging edge labels taken from the look-ahead code (Table 5.12), where the codewords are delayed by one block interval. It should be appreciated that the standard finite-state encoder given above is not always the preferred embodiment of the coding rules in practice. It is usually advantageous to translate the code table plus the look-back and look-ahead dependences directly into logical gates using a CAD package rather than minimizing the number of encoder states.

Clearly, the code of size six shown in Table 5.12 is of great academic interest only, if it were not possible to generalize it to more interesting code sizes. Indeed, the simple code presented in the preceding example can be generalized for arbitrary values of the codeword length $n > 3$. The size of the code, denoted by $N(n)$, is given by

$$\begin{aligned} N(n) &= 0, n < 0, \\ N(0) &= 1, \\ N(n) &= N(n-3) + F(n-1), n > 0, \end{aligned} \tag{5.11}$$

where the Fibonacci numbers $F(n)$ are given by

$$\begin{aligned} F(n) &= 0, n < 0, \\ F(0) &= 1, F(1) = 2, \\ F(n) &= F(n-1) + F(n-2), n > 1. \end{aligned} \tag{5.12}$$

Despite the elegance of the recursive expression, (5.11) could only be proved after a straightforward, but tedious, process of enumeration and, therefore, the proof is omitted.

Table 5.13 shows $N(n)$ as a function of the codeword length n . The size of the dk -constrained block code of the same codeword length, $F(n-1)$, is included for comparison purposes. The table reveals that the smallest rate $2/3$, ($d=1$) RLL block-decodable code has a codeword length $n=6$. For the same rate, the conventional (d, k) We also note that $N(9) = 72$, which suggests the feasibility of the construction of a rate $6/9$, ($1, k < \infty$) RLL block-decodable code and, of even more practical relevance, that $N(12) = 305$, which suggests the feasibility of the construction of a rate $8/12$, ($1, k < \infty$) RLL block code with the attractive source word length $m=8$. Both codes are, as can be seen in Table 5.13, not possible with conventional $d, k < \infty$ -constrained block codes of the same length. block code requires a codeword length of at least 18 (see Section 5.3.1, page 100).

Table 5.13: Code size $N(n)$ of $(1, \infty)$ RLL block codes of codeword length n along with the code size, $F(n-1)$, of conventional (d, ∞) codes of the same codeword length.

n	$N(n)$	$F(n-1)$
4	6	5
5	10	8
6	17	13
7	27	21
8	44	34
9	72	55
10	116	89
11	188	144
12	305	233

Numerous RLL block-decodable codes can be constructed for runlength parameters and word lengths m and n for which no conventional (d, k) block-decodable codes could be constructed (see e.g. Immink [149] and Hollmann [136]). A brief survey of RLL block-decodable codes of practical interest is listed in Table 5.14. For comparison purposes we have tabulated in parentheses the shortest possible codeword length of conventional block-decodable (d, k) codes of the same rate and runlength parameters, which were found using the method presented in Section 5.3.1.

Table 5.14: Survey of RLL block-decodable codes. The number in parentheses denotes the codeword length of a (d, k) block-decodable code with the same specifications as the RLL block-decodable code.

d	k	m	n	
0	1	4	6	(18)
0	2	5	6	(12)
1	∞	4	6	(18)
1	10	6	9	(21)
1	9	8	12	(21)
2	∞	3	6	(14)
2	8	8	16	(22)
3	10	6	15	(20)

The rate 8/12, (1,9) and the rate 8/16, (2,8) RLL block-decodable codes are of specific interest as they are well adapted to byte-formatted storage systems. A beautiful rate 8/12, (1,8) block-decodable code was constructed

by Hollmann [136]. The rate $8/12$, $(1, 9)$ code listed in Table 5.14, requires 358 encoder states. This number may, at first sight, look prohibitively large, but this is merely the result of the standard finite-state machine description chosen. Given the finite-state encoder, we can easily construct a code book as depicted in Table 5.12, and this code book, in turn, can, using the look-back and look-ahead features, be implemented in a straightforward manner. The required hardware is well within the reach of modern LSI. The code is much more complex than the industry standard rate $2/3$, $(1, 7)$ code, (see Section 7.4.1, page 171), but any coding scheme is merely a part of a larger system and its cost must be in proportion to its importance within that system.

5.8 Almost block-decodable codes

The technique presented in this section is broadly similar to "Constructions 1 and 2". An essential, and very useful, property of these constructions, which they have in common with the constructions to be discussed here, is that source words have a one-to-one relationship with finite-length (dk) sequences that have additional constraints on the number of leading and trailing 'zero's. After translating the source words into (dk) sequences, these sequences are multiplexed with $\beta = d$ bits, called *merging bits*, which are needed to preserve the predefined runlength constraints. Constructions 1 and 2 have the engineering virtues that

- the translation of source words into (dk) constrained codewords can be accomplished with a single look-up table,
- the cascading of codewords can be done with a simple merging rule involving $\beta = d$ merging bits,
- their structure permits enumerative coding techniques, see Chapter 6, to simplify encoding and decoder hardware, which is a particularly desirable feature when the codeword length is relatively large, and
- they are simple to understand.

The constructions, to be discussed in this section, have the same virtues, but less than d merging bits are needed for cascading the (dk) sequences. An essential feature of these codes is look-ahead, that is, during the encoding process, the codeword is not only a function of the history, but it is also a function of the upcoming data. The merging operation affects both the merging bits and a few bits of the adjacent (dk) sequences. As a result, in contrast with Construction 1 and 2, the codes presented in this section are almost-block-decodable as decoding of the sequence can be accomplished by

observing (part of) the received codeword plus a small part of the previous codeword. In two constructions, called Construction 3 and 4, a codeword and, in addition, one bit of the previous codeword have to be observed for proper decoding, while in Construction 5, in addition three bits of the previous codeword must be observed. As a result, minor error propagation may occur during decoding.

We commence our exposition with a description of a simple code, called *Three Position Modulation (3PM)*, which has been used in disk systems. The various constructions to follow are generalizations of the 3PM code.

5.8.1 Three Position Modulation (3PM) code

The basic parameters of the 3PM (Three Position Modulation) code, which was invented by Jacoby [172, 173] are $n = 6$, $d = 2$, $k = 11$, and $R = 1/2$. The encoding mechanism of the 3PM code is similar to block encoding and decoding with one extra merging rule, which can easily be understood by looking at Table 5.15.

Table 5.15: Basic coding table 3PM code.

<i>Data</i>	<i>Code</i>
000	000010
001	000100
010	010000
011	010010
100	001000
101	100000
110	100010
111	100100

As we may notice, the right-hand boundary bit at position 6, the merging bit, is set to 'zero'. If the juxtaposition of codewords will offend the $d = 2$ condition, that is, if both the symbol at position 5 of the present codeword and the symbol at position 1 of the next codeword are 'one', the merging bit is set to 'one' and the symbols at positions 1 and 5 are set to 'zero'. At the receiver site, we can observe that such a modification has been made, and an action to undo it is easily done. Note that there are 129 codewords of length 13 (including one merging bit), so that a rate $7/13$ code is immediately possible with the above 3PM construction. Kim [204] has been granted a U.S. Patent on an embodiment of a rate $7/13$, $(2,25)$ code, which is based on 3PM. An example of a rate $4/8$, $(2,9)$ RLL code with additional merging rules to limit the k constraint was given by Tanaka [316]. Tanaka cleverly

observed that if the merging bit equals 'one' that it cannot be followed by the string '001' or preceded by the string '100'. This property can be used to reduce the k -constraint, or to reduce the 'dc-content' (see Chapter 11). The above simple examples describing the, $d = 2$, 3PM code are easily generalized to other values of d .

5.8.2 Constructions 3, 4, and 5

The source words are represented by $(dklr)$ sequences of a length $n - \beta$, which are cascaded using $\beta = d - 1$, $d > 1$ merging bits. For larger values of d , namely $d \geq 5$, it is possible to reduce the number of merging bits to $\beta = d - 2$. For even larger values of d we can further reduce the number of merging bits (see Section 5.8.3, page 131). We start with Construction 3 using $\beta = d - 1$, $d > 1$, merging bits.

Construction 3: Choose the parameters d , $d \geq 2$, k , and n , and let the number of merging bits be $\beta = d - 1$. We have to choose k so that $k - \lceil d/2 \rceil > d$. The parameters r and l are specified by $l + r = k - \beta$. For reasons of symmetry, the set of permitted words is maximized if $r = \lfloor (k - \beta)/2 \rfloor$ and $l = (k - \beta - r)$. Then the $(d, k - \lceil d/2 \rceil, l, r)$ sequences of a length $n - \beta$, excluding the words $0^{n-\beta}$ and $0^{n-\beta-1}1$, can be cascaded with a simple rule. Note that these two words are excluded to insure that the juxtaposition of more than two codewords does not produce a potential problem. This condition is merely imposed for clerical convenience as codes can indeed be built with the excluded words, see Section 5.8.1, or the literature, for example, [316, 238].

The merging operation runs as follows. Set the β merging bits to 'zero'. If both the last bit of the current word and the first bit of the next word are equal to 'one', i.e., the d constraint will be violated after a concatenation, then take the following action:

set both these bits equal to 'zero',

set the merging bit at merging bit position $a = \lceil d/2 \rceil$ equal to 'one.'

The merging bit at position a is termed the *pivot bit*, and the operation described above is called a *pivoting operation*. The pivoting operation provides that the two 'one's that are too close together, will be replaced by a single 'one' at merging bit position a . As a result of the pivoting operation, the most left 'one' of the next word is 'shifted' $\lceil d/2 \rceil$ positions, which is the reason for restricting the codebook to $(dklr)$ sequences with a maximum runlength equal to $k - \lceil d/2 \rceil$. Generally speaking, this constraint is too tight, as it is in force for the entire word. It is easily seen that it is sufficient to exclude, for d is even, from the set of $(dklr)$ sequences those sequences that start with the string 10^p , $p > k - d/2$, or end with 0^p1 ,

$p > k - d/2$. For odd d , it is sufficient to bar those sequences that start with 10^p , $p > k - (d - 1)/2$, or end with 0^p1 , $p > k - (d + 1)/2$. The reason for restricting the code set to $(d, k - \lceil d/2 \rceil, l, r)$ sequences is that this set has the desirable property that it consists of $(dklr)$ sequences only, which permit enumerative encoding and decoding. If the usage of enumerative coding is not contemplated, then in certain instances a more efficient code can be designed by the less restrictive conditions.

In contrast with both Constructions 1 and 2, where during the decoding operation, all merging bits are skipped by the decoder, decoding is done here by observing $n - \beta$ bits of the received word *plus* the pivot bits preceding and succeeding the actual codeword. That is, a detection window of in total $n - d + 3$ bits is required for undoing the modifications made. A logic array, which embodies the inverse of the encoding function, can readily be used for translating the $n - d + 3$ bits into the recovered m -bit source symbols. If enumerative coding is employed, then decoding is done in two steps. With a small logic array the $n - d + 3$ bits can be uniquely reconstituted into the corresponding $(n - \beta)$ -bit $(dklr)$ sequence, which, in turn, can be translated, using enumerative coding, into the recovered m -bit source tuples. It should be appreciated that all merging bits, except the pivot bit, can be skipped.

As the code format involves the translation of source words into $(dklr)$ sequences, the code permits enumerative coding techniques to simplify encoding and decoder hardware. Note that the removal of the two words 0^{n-d+1} and $0^{n-d}1$ does not destroy the simple lexicographical ordering of the $(dklr)$ sequences. Enumerative encoding and decoding is particularly attractive when the codeword length is relatively large. A description of enumerative coding of $(dklr)$ sequences can be found in Section 6.2.

The efficiency of the 3PM design is easily seen from the following argument. For large n we have $N_d(n) \approx A2^{C(d,\infty)n}$, where A is independent of n . It is clear from the above that the increase in code size is approximately $2^{C(d,\infty)}$. For $d = 2$, for example, we have $2^{C(d,\infty)} \approx 1.47$, and we conclude that the code size increases by almost 50%. Table 6.2, page 148, gives an indication of the relative efficiency of the various block construction when the codewords are (very) long.

In the previous constructions, one of the β merging bits is used as a pivot bit, and the remaining $\beta - 1$ merging bits are set to 'zero'. In the next construction, called Construction 4, for $d \geq 3$, these "unused" merging bits can be exploited, in line with Construction 2, for constraining the maximum runlength. The rate effectiveness of Construction 3 can thus be improved with a slightly more complex merging rule.

Construction 4: Choose the parameters d , $d \geq 3$, k , and n , $k > 2d$, and let the number of merging bits be $\beta = d - 1$. We have to choose k so that

$k - \lceil d/2 \rceil > d$. The parameters r and l are given by $r = k - d - 1$ and $l = k - d$.

The merging bits are functions of the number of 'zero's preceding and succeeding them. In order to define this function, let s be the number of trailing 'zero's of the current $(dklr)$ sequence, and let t be the number of leading 'zero's of the next $(dklr)$ sequence. Then $(d, k - \lceil d/2 \rceil, l, r)$ sequences of length $n - \beta$, excluding the all-0's word, can be cascaded by choosing the merging bits as listed in Table 5.16.

Table 5.16: Merging rule of $(dklr)$ sequences.

s, t	Merging bits
$s = t = 0$	$0^{a-1}10^{d-a-1}$
$1 \leq s + t \leq k - d + 1$	0^{d-1}
$s + t > k - d + 1$	
if $s < d \cap d - s \neq a - 1$	$0^{d-s}10^{s-2}$
if $s < d \cap d - s = a - 1$	0^a10^{d-a-2}
if $s \geq d > 3$	10^{d-2}
if $s \geq d = 3$	01

It should be noted that the position a of the pivot bit is given by $a = \lfloor d/2 \rfloor$. If the last bit of the current word and the first bit of the next word are both equal to 'one', i.e., $s = t = 0$ then a pivot operation is performed, i.e., both the last bit of the actual codeword and the first bit of the next codeword are set to 'zero'. Essentially, all $\beta = d - 1$ merging bits are set to 'zero', unless the d - or k -constraint will be violated. In the former case the pivot bit at merging bit position a is set to 'one', and in the latter case a properly chosen merging bit, not being the pivot bit at position a , is set to 'one'. Table 5.16 displays a fixed rule, but it should be understood that there is certain degree of freedom to position the 'one'. The degree of freedom offered can, if required, be utilized for specific purposes, such as synchronization or the minimization of the signal power at given frequencies. Decoding is done, as described in Construction 3, by observing $n - \beta$ bits of the received word plus the pivot bits preceding and succeeding them. All merging bits except the pivot bit can be skipped. In the previous construction, one of the $\beta = d - 1$ merging bits is used as pivot bit, and we exploited, for $d > 2$, the surplus of merging bits for constraining the maximum runlength. The next construction, which is valid for $d > 4$, employs $\beta = d - 2$ merging bits; three of them are pivot bits.

Construction 5: Choose the parameters d , $d \geq 5$, k , and n , and let the

number of merging bits be $\beta = d-2$. We have to choose k so that $k - \lfloor d/2 \rfloor > d$. The parameters r and l are given by $r = \lfloor (k - \beta)/2 \rfloor$ and $l = (k - r - \beta)$. Then the $(d, k - \lfloor d/2 \rfloor, l, r)$ sequences of a length $n - \beta$, excluding the words $0^{n-\beta}$, $0^{n-\beta-1}1$, and $0^{n-\beta-2}10$, can be cascaded with a simple rule. In this construction, three of the β merging bit act as pivot bits. The positions of the pivot bits are denoted by a_1, a_2 , and a_3 . The basic idea is that the pivot bits can be set to 'one' if the d -constraint will be violated during the cascading operation. The pivot bits are chosen, for reasons of symmetry, as close to the middle of the β merging bits as possible.

Let s be the number of trailing 'zero's of the current $(dklr)$ sequence, and let t be the number of leading 'zero's of the next $(dklr)$ sequence. Then, as can easily be verified, there are three combinations where during a cascading operation the minimum runlength could be offended, namely, $(s = t = 0)$, $(s = 1, t = 0)$, and $(s = 0, t = 1)$. If the minimum runlength is in danger of becoming too short, a violation of the d -constraint can be circumvented by executing a pivoting operation involving one of the three pivot bits. If, for example, $s = t = 0$, then pivoting is performed with the pivot bit at position a_1 (by setting it to 'one' and by setting the 'one's at the beginning and end of the adjoining $(dklr)$ sequences to 'zero'). If $(s = 1, t = 0)$, the pivoting operation is carried out with the pivot bit at position a_2 , etc. The possible execution of one of the three distinct pivoting operations can be uniquely decoded by observing the pivot bits, and an action, at the receiver's end, undertaken to restore the original $(dklr)$ sequences is easily performed. Thereafter, decoding is as usual: all merging bits, except the three pivot bits, can be skipped.

5.8.3 Generalized construction

In this subsection, taken from [203], the above constructions are generalized to large values of d . To that end, let s be the number of trailing 'zero's of the current (d) sequence, and let t be the number of leading 'zero's of the upcoming (d) sequence. There is no violation of the d constraint if $t + \beta + s \geq d$. Then the merging bits are set to 'zero' and the n -bit codeword plus the β merging bits are transmitted. If a d -constraint violation would occur, i.e. if $t + \beta + s < d$, we will perform a special operation, which will be described shortly. The number of combinations, where a violation occurs is simply

$$\frac{(d - \beta + 1)(d - \beta)}{2}, \quad d \leq 2, \beta \leq 1. \quad (5.13)$$

A violation of the d -constraint can be circumvented by executing an operation involving the merging bits and the two 'one's next to the merging bits that trespass the d -constraint. The two trespassing 'one's are set to 'zero' and a judiciously chosen merging bit is set to 'one'. Let the positions of

the merging bits be denoted by a_1, \dots, a_β . If, for example, $s = t = 0$, we set the merging bit at position a_1 equal to 'one' and set the 'one's at the beginning and end of the adjoining (d) sequences to 'zero'. If ($s = 1, t = 0$), the merging bit at position a_2 is set to 'zero' etc. Clearly, if

$$\frac{(d - \beta + 1)(d - \beta)}{2} \leq \beta \quad (5.14)$$

it is possible to uniquely denote the trespassing combinations by the β -tuple $0^i 10^{\beta-i-1}$, $0 \leq i \leq \beta - 1$, where 0^i denotes a string of i consecutive 'zero's. From (5.14), we can easily compute the value of the number of merging bits, β , as a function of d . Table 5.17 gives the results. The possible execution of one of the β distinct merging operations can be uniquely decoded by observing the merging bits, and an action, at the receiver's end, undertaken to restore the original (d) sequences is easily performed.

Table 5.17: The number of merging bits β as a function of d .

d	β
2	1
5	3
9	6
14	10
20	15

5.8.4 Results and comparisons

The algorithm outlined in the previous sections has been implemented and successfully applied to find almost-block-decodable codes. Our effort has been focused on the finding of good candidate codes that are byte oriented. Selected results of our investigations are listed in Table 5.18. The code efficiency η is defined as $\eta = R/C(d, k)$. It can be seen that the efficiency of the codes is better than 90%. Of particular interest in this respect is the (2,16), rate 8/15 code achieving an efficiency of 97 %. Another code of practical interest is the byte-oriented (2,9), rate 8/16, code found by application of Construction 3. This code has a slightly smaller maximum runlength than the rate (2,10), rate 8/16, block-decodable code that can be established with Construction 2.

A comparison of Tables 5.18 and 5.10, page 110, shows that the rate efficiency of the codes constructed with Constructions 3, 4, and, 5 are a few percent better than that of codes built with Construction 2. We should keep in mind that the codes listed in Table 5.10 are block decodable, while the

new codes listed in Table 5.18 may suffer from a slight error propagation. It is the difficult task of the system architect to seek a trade-off between the smaller maximum runlength and increased risk of error propagation of the former code against the larger maximum runlength and absence of error propagation of the latter code. It is difficult to give an answer to this question as it depends on a number of factors peculiar to a particular usage.

Table 5.18: Codes based on Constructions 3, 4, and 5.

d	k	n	R	$C(d, k)$	$\eta = R/C(d, k)$	Construction
2	16	15	8/15	0.551	0.97	3
2	9	16	8/16	0.537	0.93	3
3	11	19	8/19	0.464	0.91	4
4	14	22	8/22	0.397	0.92	4
5	20	24	8/24	0.359	0.93	5

5.9 Appendix: Generating functions

A very useful method for counting the number of certain constrained sequences is offered by *generating functions*. An extensive treatment of this topic can be found in the textbook by Riordan [290]. The idea is to associate with each sequence of numbers a function of a dummy variable in such a way that common operation on sequences correspond to common operations on the corresponding functions. Let the sequence of numbers be c_0, c_1, c_2, \dots . We associate with the sequence of numbers a *generating function*, or *generating series*, denoted by $T(x)$, and defined by the power series

$$T(x) = c_0 + c_1x + c_2x^2 + \dots$$

The variable x is a dummy variable and the generating function is a formal sum, so questions such as convergence are irrelevant. For instance, an infinite sequence of the exponentials of two,

$$1, 2, 4, 8, 16, 32, \dots$$

can be represented by the generating function

$$T(x) = 1 + 2x + 4x^2 + 8x^3 + 16x^4 + 32x^5 + \dots,$$

which can be written in a simple closed form, namely

$$T(x) = \frac{1}{1 - 2x}.$$

Typically, we have a sequence c_i such that c_i is the number of objects in a certain set that have a "value" of some sort equal to i . The relevance in combinatorics of generating functions may become apparent with a small example based on Polya's illuminating paper [286].

Suppose we have a pile of nickels (5 cent), dimes (10 cent), and quarters (25 cent), and we wish to make change for, say, a dollar. In how many ways can we do this? The number of ways to make change for a dollar is the coefficient of x^{100} in the product of the power series

$$(1 + x^5 + x^{10} + \dots)(1 + x^{10} + x^{20} + \dots)(1 + x^{25} + x^{50} + \dots).$$

The above power series can be succinctly written as

$$\frac{1}{(1 - x^5)(1 - x^{10})(1 - x^{25})}.$$

Using a mathematical toolbox we can generate the Taylor series of the above expression, and find

$$\sum_{i=0}^{\infty} c_i x^i = 1 + x^5 + 2x^{10} + 3x^{20} + 4x^{25} + \dots + 29x^{100} + \dots$$

So that we conclude there are 29 ways to make change for a dollar using nickels, dimes and quarters.

In the second example, we assume there is a set of $n + 1$ non-negative integers $\{a_0, \dots, a_n\}$. Define

$$P(x) = x^{a_0} + \dots + x^{a_n}.$$

Then, the coefficient c_i of

$$\sum c_i x^i = P(x)^m$$

equals the number of ways m integers taken from $\{a_0, \dots, a_n\}$ will add to i . For example, how many combinations of four dice can we find that have a sum of 12? Using generating functions, we find that c_{12} of

$$\sum c_i x^i = (x + x^2 + \dots + x^6)^4$$

is the answer. After working out, we find $c_{12} = 125$.

The number of ways we can add to i with *any* number of integers taken from $\{a_0, a_1, \dots, a_n\}$ is given by the coefficient c_i of the generating function

$$T(x) = 1 + P(x) + P(x)^2 + P(x)^3 + \dots.$$

The series $T(x)$ can be formally written as

$$T(x) = \frac{1}{1 - P(x)}.$$

For example, how many combinations of any number of dice can we find that have a sum of 12? Using generating functions, we find that c_{12} of

$$\sum c_i x^i = \frac{1}{1 - x - x^2 - \dots - x^6}$$

is the answer. After working out, we find $c_{12} = 1936$.

Chapter 6

Enumerative coding

6.1 Introduction

In the examples of codes described in the previous chapters, it is tacitly assumed that a table is used to hold all the codewords. For the small codes discussed this is indeed a feasible solution, but specifically, when codewords are comparatively long, the method of direct look-up could easily become an engineering impossibility. The limit depends on technology and required bit rate; figures of codeword length of 14 to 16 are commonly quoted as typical present-day maxima.

However, codewords can be computed by an *algorithmic procedure*, called *enumerative encoding*, which means that there is therefore no need to store codewords in a table. The algorithm to be discussed below is very similar to the one used for the well-known conversion of decimal to binary numbers and *vice versa*. Binary to decimal conversion is done by forming the inner product of the binary vector and a vector of weights. In standard binary to decimal conversion the weights equal the powers of two. In the general case, 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 weight coefficients. For a codeword of length n , storage capacity is required for approximately n non-zero weighting coefficients, a full adder and an accumulator to store the intermediate and final results. The enumerative decoder will contain some elements, e.g. the weighting coefficients look-up table, which are identical with the ones in the encoder. Said hardware requirements have to be contrasted with look-up tables whose size grows exponentially with the codeword length in case a, conventional, non-algebraic method for encoding and decoding is used.

As the major practical stumbling block of the construction of very efficient large block codes, the look-up tables, is removed by the usage of enumerative coding, we will take a closer look at the feasibility of the con-

struction of very large codes. Note that it is implied by Shannon's capacity theorem that block codes will approach the capacity with mounting block length, but that it is not clear at all what this statement means in practice. In Section 6.3.3, we will show that, as a rule of thumb, a (d, k) block code comprising codewords of a length of a few hundred bits can realize an efficiency which is just a few tenths of a percent below channel capacity.

Besides the complexity issue of large codes, there is a second serious difficulty that could hamper its introduction, namely massive error propagation. Clearly, the longer the codewords the more decoded bits can be in error when merely a single channel bit is received erroneously. The effects of *average* error propagation will be computed in Section 6.4. It will be shown that the average error propagation can be controlled by a judicious choice of the enumeration system, which, among others, has a bearing on the rate of the code. Alternatively, a coding scheme in conjunction with error correction, specifically constructed to circumvent worst case error propagation, will be discussed in Section 6.5.

6.2 Basics of enumeration

The translation of input data into constrained sequences and *vice versa* using enumeration has a long history [52]. The first encoder using enumeration was patented by Labin & Aigrain [216] in 1951. The inventors described a mechanism for translating user data into 35 binary codewords each having 3 out of 7 'one's. It is not known to have been put to any practical use. In 1965, Kautz [196] presented an enumeration scheme for coding $(0, k)$ -constrained sequences, followed, in 1970, by a general enumeration scheme of (dk) -constrained sequences by Tang & Bahl [319]. Recent contributions to the art can be found in [323, 229]. The description of the enumeration method given below is due to Cover [66].

Let $\{0, 1\}^n$ denote the set of binary sequences of length n and let S be any (constrained) subset of $\{0, 1\}^n$. The set S can be ordered lexicographically as follows: if $\mathbf{x} = (x_1, \dots, x_n) \in S$ and $\mathbf{y} = (y_1, \dots, y_n) \in S$, then \mathbf{y} is called less than \mathbf{x} , in short, $\mathbf{y} < \mathbf{x}$, if there exists an $i, 1 \leq i \leq n$, such that $y_i < x_i$ and $x_j = y_j, 1 \leq j < i$. For example, '00101' < '01010'. The *rank* of \mathbf{x} , denoted by $i_S(\mathbf{x})$, is defined to be the position of \mathbf{x} in the lexicographical ordering of S , i.e. $i_S(\mathbf{x})$ is the number of all \mathbf{y} in S with $\mathbf{y} < \mathbf{x}$.

Let $n_s(x_1, x_2, \dots, x_u)$ be the number of elements in S for which the first u coordinates are (x_1, x_2, \dots, x_u) .

Proposition 6.1 *The rank of $\mathbf{x} \in S$ satisfies*

$$i_S(\mathbf{x}) = \sum_{j=1}^n x_j n_s(x_1, x_2, \dots, x_{j-1}, 0). \quad (6.1)$$

Proof: By definition, words with prefix $(x_1, x_2, \dots, x_{j-1}, 0)$ precede words with prefix $(x_1, x_2, \dots, x_{j-1}, 1)$. For each j such that $x_j = 1$, $n_s(x_1, x_2, \dots, x_{j-1}, 0)$ gives the number of elements of S that first differ from \mathbf{x} in the j th term and therefore have a lower lexicographic index. By adding these numbers for $j = 1, 2, \dots, n$, we eventually count all the elements in S of lower index than \mathbf{x} .

Given S and the lexicographic index I , the vector $(x_1, \dots, x_n) \in S$ with index I is obtained as follows:

1. If $I \geq n_s(0)$ set $x_1 = 1$ and set $I = I - n_s(0)$; otherwise set $x_1 = 0$.
2. For $j = 2, \dots, n$, if $I \geq n_s(x_1, x_2, \dots, x_{j-1}, 0)$ set $x_j = 1$ and set $I = I - n_s(x_1, x_2, \dots, x_{j-1}, 0)$; otherwise set $x_j = 0$.

The above operation is usually called the encoding operation. The following two examples, taken from [66], will illustrate the above theory.

Example 6.1 Let $S = \{0, 1\}^n$. Then $n_s(x_1, x_2, \dots, x_k) = 2^{n-k}$, and

$$i_S(\mathbf{x}) = \sum_{k=1}^n x_k 2^{n-k}. \quad (6.2)$$

The above summation embodies the well-known binary-to-decimal conversion algorithm, where the weights equal the powers of two. For example the word '10010' translates into the integer $16+2=18$.

Example 6.2 Let S be the subset of $\{0, 1\}^n$ containing words having w 'one's and $n - w$ 'zero's, that is

$$S = \left\{ \mathbf{x} \in \{0, 1\}^n : \sum_{i=1}^n x_i = w \right\}.$$

Then

$$n_s(x_1, x_2, \dots, x_{k-1}, 0) = \binom{n-k}{n(w, k)},$$

where

$$n(w, k) = w - \sum_{i=1}^{k-1} x_i.$$

Therefore,

$$i_S(\mathbf{x}) = \sum_{k=1}^n x_k \binom{n-k}{n(w, k)}. \quad (6.3)$$

As an example consider the 6-bit word $\mathbf{x} = '010100'$ with four 0's and two 1's. According to the above enumeration scheme, we can find the rank $i_S(\mathbf{x})$ of \mathbf{x} using the array of binomial coefficients from Pascal's triangle (see Figure 6.1).

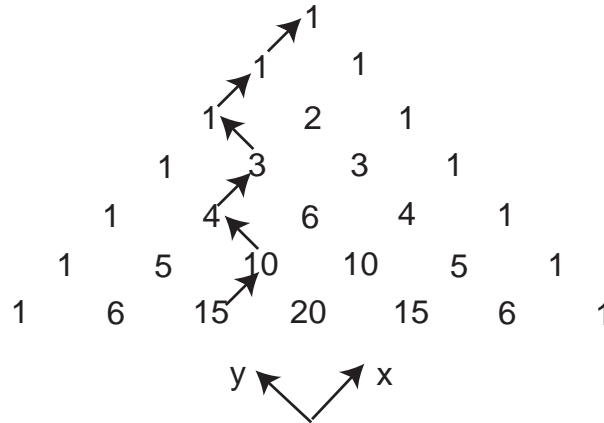


Figure 6.1: Pascal's triangle gives a representation of the enumeration scheme. The arrows indicate the path taken by the word '010100'.

Start at the lower left-hand corner of the array, at 15 in Figure 6.1. The leftmost digit of the sequence $\mathbf{x} = '010100'$ is a '0'. Move one step in the X direction, i.e. toward 10. The next digit is a '1'. Move one step in the Y direction, i.e. toward 4, and record the number at a single step in the X direction from the current starting point 3, giving $6+2=8$. The last two digits are 0's leading to no more additions. Thus the final result is $i_S(\mathbf{x}) = 8$.

The next example generalizes the previous example for a non-binary alphabet. It was shown by Datta & McLauhlin [67] and Milenkovic & Vasic [246] that the algorithm can be employed to enumerate (dk) constrained phrases.

Example 6.3 The allowable (d, k) sequences can be thought to be made of phrases, where each phrase starts with a 'one' and ending with at least d and at most k 'zero's. We will discuss codewords that can be formed by N given phrases. The set, U , of such (d, k) codewords (note that by definition such a codeword starts with a 'one') can be defined by a *phrase profile vector* $\mathbf{v} = (v_1, v_2, \dots, v_N)$. The components of \mathbf{v} take on values from the set $\{d+1, \dots, k+1\}$ and denote the N phrase lengths that will be used in a codeword. If we let n_j be the number of times j appears in \mathbf{v} , then the number of codewords that can be formed is given by

$$|U| = \frac{N!}{n_{d+1}!n_{d+2}!\dots n_{k+1}!}.$$

The codeword length (in bits) is $\sum n_j$. Let x_j , $1 \leq j \leq N$, be the j th transmitted phrase. Then, using Proposition 6.1, we have

$$n_s(x_1, x_2, \dots, x_{u-1}, 0) = \binom{(N-u)!}{G(u)},$$

where

$$G(u) = (n_{d+1} - n_{d+1}^{(u)})!(n_{d+2} - n_{d+2}^{(u)})! \dots (n_{k+1} - n_{k+1}^{(u)})!$$

and $n_j^{(u)}$ denotes the number phrases of length j in the first u phrases of the codeword. Therefore,

$$i_S(\mathbf{x}) = \sum_{u=1}^N (x_u - d - 1) \binom{(N-u)!}{G(u)}. \quad (6.4)$$

The next example shows that enumeration can readily be applied to a set of (d) sequences.

Example 6.4 Let S be the subset of (d) sequences of length n . Then $n_s(x_1, x_2, \dots, x_k) = N(n - k)$. Thus

$$i_S(\mathbf{x}) = \sum_{j=1}^n x_j N(n - j), \quad (6.5)$$

where $N(n)$ is the number of (d) sequences of length n , $n > 0$. By definition $N(0) = 1$. It is of some interest to verify the above. To that end, consider the set S of $(d = 1)$ sequences of length 4. As $N_1(0) = 1$, $N_1(1) = 2$, $N_1(2) = 3$, $N_1(3) = 5$, and $N_1(4) = 8$. For instance, $r(1001) = N_1(3) + N_1(0) = 5 + 1 = 6$. We can 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

In an implementation of the enumeration scheme, the weight coefficients $N(n)$ can be pre-calculated and stored in memory or they can be calculated 'on the fly'. In the more specific case of $(dklr)$ sequences, similar equations can be written down (see next section).

6.3 Enumeration of $(dklr)$ sequences

The general case of enumeration of $(dklr)$ sequences cannot easily be described with Proposition 6.1. The following alternative scheme can be used instead.

An alternative to Cover's enumeration scheme can be given by counting the number of elements of S that have a *higher* lexicographic index than \mathbf{x} , the *inverse rank* of \mathbf{x} .

Proposition 6.2 *The inverse rank of $\mathbf{x} \in S$ satisfies*

$$i_S^c(\mathbf{x}) = \sum_{j=1}^n \bar{x}_j n_s(x_1, x_2, \dots, x_{j-1}, 1), \quad (6.6)$$

where $\bar{x}_j = 1 - x_j$, the complement of x_j .

Proof: Similar to the one of Proposition 6.1 and therefore omitted.

In the theory below, due to Patrovics *et al.* [277], we will first give an algorithm for enumerating *dkr*-constrained sequences for which the length of the leading 'zero'-run is not constrained. From the results obtained, enumeration of (*dklr*) sequences follows easily.

First, we will introduce some notations that will facilitate the use of Proposition 6.2. Given a *dkr*-codeword \mathbf{x} , let $\mathbf{x}_j^1 = (x_1, x_2, \dots, x_{j-1}, 1)$. Denote by $N^0(i)$ the number of *dkr* constrained sequences of length i whose first element equals 1. We define the quantity $a_j(\mathbf{x})$ as the length of the trailing zero-run of the sub-vector (x_1, \dots, x_{j-1}) if it is not the all-zero sequence. Hence,

$$a_j(\mathbf{x}) = \begin{cases} \min\{(j - i - 1) : 1 \leq i < j, x_i = 1\} & \text{if } j > 1 \text{ and} \\ & (x_1, \dots, x_{j-1}) \neq \mathbf{0}; \\ d & \text{otherwise.} \end{cases}$$

With the above notation, we can write down the next proposition for enumerating (*dkr*) sequences.

Proposition 6.3 *The inverse rank of a (*dkr*) sequence \mathbf{x} of length n is*

$$i_S^c(\mathbf{x}) = \sum_{j=1}^n \delta_j(\mathbf{x}) N^0(n - j + 1),$$

where

$$\delta_j(\mathbf{x}) = \begin{cases} 1 & \text{if } x_j = 0 \text{ and } a_j(\mathbf{x}) \geq d; \\ 0 & \text{otherwise.} \end{cases}$$

Proof: If $a_j(\mathbf{x}) < d$ then \mathbf{x}_j^1 contains at least one 'one' in the first $(j - 1)$ positions, and the length of the zero-run between the last two 1s of \mathbf{x}_j^1 is less than d , violating the d constraint. Therefore, no (*dkr*) codeword begins with \mathbf{x}_j^1 , thus $n_s(\mathbf{x}_j^1) = 0$. If $a_j(\mathbf{x}) \geq d$, then \mathbf{x}_j^1 does not violate the d constraint, and so $n_s(\mathbf{x}_j^1)$ equals the number of (*dkr*) sequences beginning with \mathbf{x}_j^1 . As \mathbf{x}_j^1 ends with a "1", $n_s(\mathbf{x}_j^1)$ equals $N^0(n - j + 1)$. By noting the

above, we obtain the proposition for the enumeration of (dkr) sequences.

Clearly, (dkr) sequences having more than l leading 'zero's have lower lexicographic indexes than $(dklr)$ sequences. Therefore their inverse rank is higher. Let \mathbf{x}_l denote the $(dklr)$ codeword with the highest inverse rank in S , then the set $\{0, \dots, i_S^c(\mathbf{x}_l)\}$ of inverse ranks corresponds to the set of all $(dklr)$ codewords of a length n . Consequently, the encoding algorithm does not need the subtraction.

Proposition 6.3 provides the algorithm for decoding $(dklr)$ sequences. The encoding operation, that is, given an integer I , find the vector \mathbf{x}_l with inverse rank $I = i_S^c(\mathbf{x}_l)$ is given below.

```

 $\hat{I} := I, a := d;$ 
for  $j = 1$  to  $n$  do
  if  $\hat{I} \geq N^0(n - j + 1)$  and  $a \geq d$ 
    then  $x_j := 0, \hat{I} := \hat{I} - N^0(n - j + 1)$ 
    else if  $a < d$ 
      then  $x_j := 0$ 
      else  $x_j := 1, a := -1;$ 
   $a := a + 1;$ 
end for

```

The computation of the weights $N^0(n)$ is an exercise in combinatorics. An elegant computational method, which makes it possible to accurately approximate $N^0(n)$ for large values of n , is based on generating functions (see Section 6.3.3). Here we opt for a simple counting argument.

To that end, let $U(n)$ be the number of (dk) sequences that start and end with a '1'. Then the following recursions are immediate:

$$U(n) = \begin{cases} 0, & \text{if } n \leq 0, \\ 1, & \text{if } n = 1, \\ \sum_{i=d}^k U(n - i - 1), & \text{if } n \geq 2. \end{cases} \quad (6.7)$$

The number of (dkr) -constrained sequences of length n , $N^0(n)$, is simply

$$N^0(n) = \sum_{i=0}^r U(n - i). \quad (6.8)$$

For $n > d + k$ we may verify the following recursion:

$$N^0(n) = \sum_{i=d+1}^{k+1} N^0(n - i). \quad (6.9)$$

6.3.1 Enumeration using floating-point arithmetic

The above coding and decoding algorithms lend themselves 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 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 hardware for implementing the enumeration process comprises a (binary) adder, a subtractor, a comparator, and a look-up table of the pre-computed set of weights $\{N^0(i)\}$, $1 \leq i \leq n$. The binary fixed-point representation of the weights requires Rn bits, where R , $0 < R < 1$, is a constant. Therefore storage proportional to n^2 is required, which is prohibitive for the long codewords we have in mind. In the sequel, we will develop an enumeration method where the weights are specified in finite-precision floating-point notation.

The floating-point notation is convenient for representing numbers that differ many orders of magnitude. In this notation, each weight is represented by s bits. As s grows with $p + \log n$, p a positive integer, the hardware required for weight storage and so on grows with $n(p + \log n)$ with the codeword length n . The price tag attached to the finite-precision representation of the weights is that it will entail a (small) loss in code rate. A quantitative trade-off between the precision of the number representation and concomitant code redundancy will be detailed in the next section.

We employ a two-part radix-2 representation

$$I = (m, e)$$

to express the weight

$$I = m \times 2^e,$$

where I , m , and e are non-negative integers. The two components m and e are usually called *mantissa* and *exponent* of the integer I , respectively. The translation of a weight into (m, e) is easily accomplished. From our context it is easily seen that the exponent e of the weights $N^0(i)$, $1 \leq i \leq n$, can be represented by at most $\lceil \log_2 Rn \rceil$ bits. With the following procedure we ensure that the mantissa m is represented by p bits. Each weight can thus be represented by $s = \lceil \log_2 Rn \rceil + p$ bits. There are various techniques such as rounding and truncation for translating fixed-point representations into floating-point representations. We choose here for a truncation operation as it is, as will be shown later, compatible with the basic enumeration algorithms.

Let I be the short-hand notation of one of the weights $N^0(i)$. It is well known that the non-negative integer I , $I < 2^v$, can be uniquely represented by a binary v -tuple $\mathbf{x} = (x_{v-1}, \dots, x_0)$, where

$$I = \sum_{i=0}^{v-1} x_i 2^i.$$

The binary v -tuple \mathbf{x} is called the binary fixed-point representation of I . Let

$$u = \lfloor \log_2 I \rfloor$$

be the position of the leading 'one' element of \mathbf{x} . Then the p -bit truncated decimal representation of I , denoted by $\lfloor I \rfloor_p$,

$$\lfloor I \rfloor_p = \lfloor 2^{-(u+1-p)} I \rfloor 2^{u+1-p}, \quad (6.10)$$

can be represented in binary floating-point representation whose mantissa requires at most p non-zero bits.

If the above finite-precision arithmetic is used in the enumeration algorithms, we must modify the set of weights developed in the previous section. To that end, let $\hat{N}^0(i)$ denote the number of (dkr) sequences of length i starting with a '1' that can be encoded with p -bit mantissa representation, then

$$\hat{N}^0(i) = \begin{cases} N^0(i), & i \leq p_1 \\ \lfloor \sum_{j=d+1}^{k+1} \hat{N}^0(i-j) \rfloor_p, & i > p_1. \end{cases} \quad (6.11)$$

For convenience, it is assumed that $N^0(p_1) < 2^p$, i.e. the smallest weights $N^0(i)$, $1 < i \leq p_1$, can be represented by a mantissa of p bits, and $p_1 \geq k+d$, i.e. the range where the homogeneous recursion is valid. The encoding and decoding algorithms developed in the previous section can be employed directly by using the 'truncated' coefficients $\hat{N}^0(i)$ in lieu of $N^0(i)$. The enumeration algorithm itself remains unchanged. The effect on the set of codewords will be that recursively the $N^0(i) - \lfloor N^0(i) \rfloor_p$ highest ranking (dkr) words of length i are discarded from the set of all lexicographically ordered dkr sequences starting with a '1'.

6.3.2 Effects of floating point arithmetic

Using finite precision of the weights representation (truncation) will result in coding loss as available $dklr$ words must be discarded. In this subsection we will study quantitatively the coding loss.

The number of (dkr) sequences of a length n that start with a '1', $N^0(n)$, $n > k+d$, satisfies the recurrence relation (see (6.9))

$$N^0(n) = \sum_{i=d+1}^{k+1} N^0(n-i).$$

The number of (dkr) sequences grows exponentially with n , the growth factor being $\lambda = 2^{C(d,k)}$. Recall that for sufficiently large n , the number of (dkr) sequences of length n , $\hat{N}^0(n)$, that can be encoded using a p -bit mantissa representation is

$$\hat{N}^0(n) = \lfloor \sum_{i=d+1}^{k+1} \hat{N}^0(n-i) \rfloor_p. \quad (6.12)$$

Conceptually, the computation of the growth factor $\hat{\lambda}$ of $\hat{N}^0(n)$ and the capacity $\hat{C}(d,k) = \log_2 \hat{\lambda}$ becomes very simple if we note that the sequence behavior of the p -bit mantissa of $\hat{N}^0(n)$ versus n can be described in terms of an autonomous finite-state machine. The characteristic functions describing the sequence behavior of the finite-state machine are implied by the recursive equation (6.12). After a little thought the following proposition will become clear.

Proposition 6.4 *The capacity $\hat{C}(d,k)$ is rational.*

Proof: From the theory of feedback registers [112] we know that the sequence of the mantissa of $\hat{N}^0(n)$ will ultimately become (and remain) *periodic*. That is, there are integers h and f such that for all sufficiently large n

$$\hat{N}^0(n)2^h = \hat{N}^0(n+f). \quad (6.13)$$

In other words, per cycle period of length f the number of sequences increases with a fixed factor, which is equal to a power of two, 2^h . From the above it is immediate that

$$\hat{C}(d,k) = \frac{h}{f}, \quad (6.14)$$

which concludes the proof.

The theory of feedback registers [112] stipulates that the cycle period must be smaller than $2^{p(k+2)}$. As this number is huge in the range of parameters of practical interest, we are inclined to believe that Proposition 6.4 is not of great practical interest. However, results of a computer search, which are listed in Table 6.1, reveal relatively small cycle periods (small denominators) are surprisingly frequent. The above method for computing the capacity $\hat{C}(d,k)$ has the virtues of precision and efficiency, but it does not provide a very useful relationship between p and the capacity loss $C(d,k) - \hat{C}(d,k)$. In the working range $d = 1, \dots, 3, k \gg d$, the loss in capacity resulting from the truncation of the weights can be simply approximated by [154]

$$C(d,k) - \hat{C}(d,k) \approx 2^{-(p+2)}. \quad (6.15)$$

Table 6.1: Capacity $\hat{C}(d, k)$ for selected values of d , k , and p .

d	k	$C(d, k)$	p	$\hat{C}(d, k)$	$C(d, k) - \hat{C}(d, k)$
0	5	0.98811	7	62/63	0.00398
1	7	0.67929	8	40/59	0.00132
1	12	0.69299	9	9/13	0.00068
2	7	0.51737	9	61/118	0.00042
2	12	0.54711	7	6/11	0.00166
2	15	0.55011	11	11/20	0.00011
3	12	0.45555	7	5/11	0.00010

6.3.3 Very long block codes

In this section, we will take a look at the asymptotic behavior of the efficiency of block codes based on Construction 1, 2, and 3 (see Section 5.4, page 106) as a function of the word length. The number of $(dklr)$ words, $N_c(n)$, of length n , available in one of the three constructions equals the coefficient a_n of the following generating function (see Section 5.4.1, page 107):

$$\sum a_i x^i = \frac{q(x)}{p(x)} = \frac{x^{\beta+1}(1-x^{l+1})(1-x^{r+1})}{(1-x)(1-x-x^{d+1}+x^{k+2})} \quad (6.16)$$

The parameters β , l , and r are stipulated by the construction recipe c_i , $i = 1, 2, 3$, where c_i denotes Construction 1, 2, or 3, respectively. For large codeword length n the number of codewords can be approximated by

$$N_c(n) \approx A_c \lambda^n, \quad (6.17)$$

where λ is the largest real root of the characteristic equation

$$z^{k+2} - z^{k+1} - z^{k-d+1} + 1 = 0 \quad (6.18)$$

and the constant A_c equals (see Appendix 6.6, page 157)

$$A_c = -\lambda \frac{q(1/\lambda)}{p'(1/\lambda)}. \quad (6.19)$$

The constant A_c depends on the parameters l , r , and β , which, in turn, are implied by the construction chosen. Note that the channel capacity $C(d, k)$ satisfies

$$C(d, k) = \log_2 \lambda.$$

Forsberg [86] found after evaluating many examples by computer experiments that the accuracy of (6.17) is surprisingly good. The rate of an implemented code is

$$R = \frac{1}{n} \lfloor \log_2 N_c(n) \rfloor. \quad (6.20)$$

The difference between capacity and rate of the code satisfies

$$\frac{1}{n}(-1 + \log_2 A_c) \leq R - C(d, k) \leq \frac{1}{n} \log_2 A_c. \quad (6.21)$$

In Table 6.2 we have collected results of computations.

Table 6.2: Constant $\log_2 A_c$ for selected values of d and k . The terms $C1$, $C2$, and $C3$ are short for Constructions 1, 2, and 3.

d	k	$C(d, k)$	$\log_2 A_{c1}$	$\log_2 A_{c2}$	$\log_2 A_{c3}$
1	3	0.551	-1.062	-0.357	
1	7	0.679	-0.771	-0.404	
2	7	0.517	-1.265	-0.602	-0.375
2	10	0.542	-1.039	-0.661	-0.332
3	10	0.446	-1.393	-0.807	-0.680

From Table 6.2 we conclude that for the parameters of most practical interest the constant $\log_2 A_c$ is in the range $-0.5, \dots, -0.35$. Thus a codeword length of approximately $n = 256$ suffices to approach capacity to within 0.2-0.5%.

As an example, we have computed $C(d = 2, k = 15) - R$ as a function of the codeword length n . Results of computations are shown in Figure 6.2. The diagram shows points for byte-oriented codes, i.e., codes whose source word length $\log_2 \lfloor N_c(n) \rfloor$ is a multiple of eight. We restricted ourselves to byte-oriented codes to fit standard memory and code hardware. The "raggedness" of the curve is caused by the truncation to the nearest power of two.

The combined effects of the truncation and the finite codeword length on the achievable rate can be conveniently approximated by

$$C(d, k) - R \approx \frac{1}{2n} + 2^{-(p+2)}. \quad (6.22)$$

If both design parameters are chosen such that their relative effect on the rate loss is equal, i.e. $2n = 2^{p+2}$, then we find the following fundamental relationship between hardware (storage) requirements and efficiency of a (long) dk code:

$$C(d, k) - R \approx \frac{1}{n}. \quad (6.23)$$

Preferably a ROM is used as a look-up table. Then the number of ROM output bits, p , is one less than the number of input bits, $\lceil \log_2 n \rceil$. To some extent, the above results do not depend on the specified runlength parameters d and k . As a rule of thumb, it can safely be concluded that a design (in the main case of practical interest, where $d < 3$) of any (dk) -constrained code whose rate is only 0.1% below capacity is possible with hardware of a ROM look-up table having 10 input and 9 output bits. In other words, with a hardware of 1 kByte ROM one can construct a block code whose rate is marginally less than channel capacity.

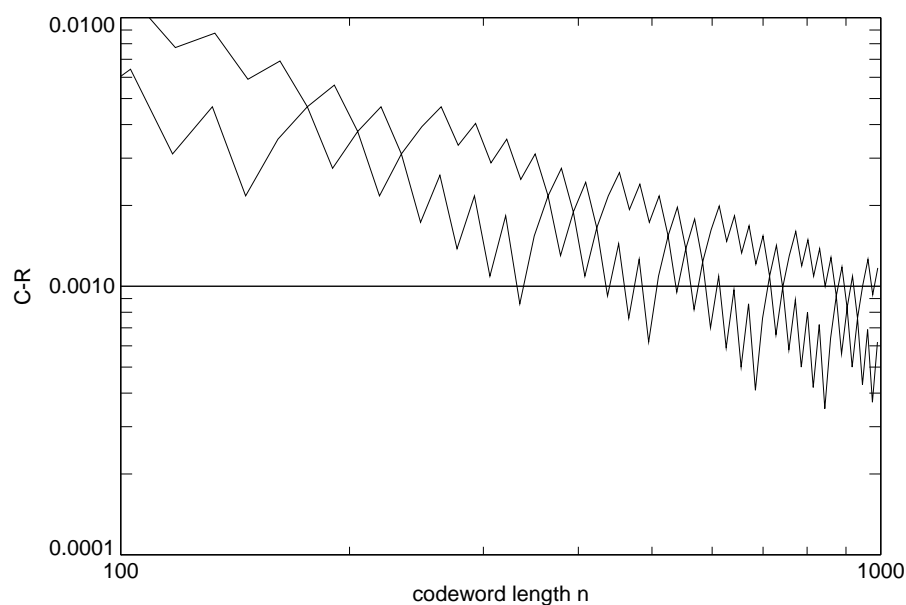


Figure 6.2: Redundancy $C - R$ versus codeword length of codes based on Construction 2 (upper curve) and Construction 3 (lower curve). The runlength parameters are $d = 2$ and $k = 15$. The channel capacity is $C(2, 15) = 0.5501$. Taken from [154]

In the above analysis it is tacitly assumed that storage is only needed for the mantissa of the weights. The exponent of the weights must also be stored, but as this additional storage is proportional to $\log n$, it is neglected. It should be appreciated that often a significant saving in storage hardware can simply be realized by noting that the mantissa of the weights ultimately becomes periodic.

By a judicious choice of k and p we may attempt to reduce the hardware requirements by minimizing the cycle period. Table 6.1 on page 147 reveals that short cycle periods (and thus small storage hardware) are possible for values of d and k of practical interest. How to systematically approach the

minimization of the cycle period is not clear yet. The effectiveness of the new technique will be illustrated by a worked example.

6.3.4 Application to (k) -constrained sequences

The capacity $C(0, k)$ equals $\log_2(\lambda)$, where λ (see Section 4.3) is the largest root of

$$x^{k+2} - 2x^{k+1} + 1 = 0. \quad (6.24)$$

For sufficiently large k , we derive

$$\lambda \approx 2\left(1 - \frac{1}{2^{k+2}}\right),$$

so that

$$C(0, k) \approx 1 - \frac{1}{\ln 2} 2^{-k-2}, \quad k \gg 1. \quad (6.25)$$

Table 6.3 lists the redundancy $1 - C(0, k)$ versus the runlength parameter k .

Table 6.3: Redundancy $1 - C(0, k)$ versus k .

k	$1 - C(0, k)$
1	0.30576
2	0.12085
3	0.05322
4	0.02477
5	0.01189
6	0.00581
7	0.00287

In Table 6.4 we have listed the maximum codeword length \hat{n} , and redundancy $1 - 1/\hat{n}$ as a function of the maximum runlength parameter k when the coefficients are not truncated. In order to maximize the number of words, we have set $r = \lceil k/2 \rceil$ and $l = k - r$ (see Section 5.4.2, page 108). We see that the redundancy required is very close to the bound listed in Table 6.3.

Table 6.4: Maximum codeword length \hat{n} for which a rate $1 - 1/\hat{n}$, $(0, k)$ code can be constructed.

k	\hat{n}
4	31
5	67
6	148
7	310
8	649

Table 6.5: Maximum codeword length \hat{n} for which a rate $1 - 1/\hat{n}$ $(0, k)$ code, $p = k + 2$, can be constructed.

k	\hat{n}
4	26
5	54
6	112
7	232
8	474

The effect of weight truncation can be seen in Table 6.5, where we have listed the maximum codeword length \hat{n} and redundancy $1 - 1/\hat{n}$ as a function of the runlength parameter k . The redundancy is only slightly larger than in the case when a full representation of the weights (see Table 6.4) is used.

The mantissa of the weights $\hat{N}^0(i)$ can be represented by at most $p > k$ bits if we use

$$\hat{N}^0(i) = \begin{cases} 0, & i \leq 0, \\ 2^{i-1}, & i = 1, \dots, r+1, \\ \sum_{j=i-1-k}^{i-1} \hat{N}^0(j), & i = r+2, \dots, p+1, \\ 2^{i-p-1} \lfloor 2^{p-i+1} \sum_{j=i-1-k}^{i-1} \hat{N}^0(j) \rfloor & i > p+1. \end{cases} \quad (6.26)$$

In the special case, $p = k + 2$, we find that mantissa and exponent of the floating-point representation of the weights are simple functions:

$$\hat{N}^0(i) = \begin{cases} 2^{i-1}, & i = 1, \dots, r+1, \\ (2^{r+1} - 1)2^{i-r-2}, & i = r+2, \dots, k+2, \\ (a_0 - i)2^{i-k-3}, & k+2 < i \leq i_1, \end{cases} \quad (6.27)$$

where

$$\begin{aligned} a_0 &= (2^{r+1} - 1)2^{k-r+1} + k + 2, \\ i_1 &= a_0 - 2^{k+1}. \end{aligned}$$

The expressions for $\hat{N}^0(i)$, $i > i_1$, are too involved and therefore omitted.

6.4 Error propagation

As we have seen in the previous section, the usage of long codewords makes it possible to approach a code rate which is arbitrarily close to Shannon's noiseless capacity of the constrained channel. Single channel bit errors may result in error propagation that could corrupt the entire data in the decoded word, and, of course, the longer the codeword the greater the number of data symbols affected.

In this section, a presentation is given of the effects of error propagation [163]. It is assumed that a binary source word, \mathbf{b} is translated into a binary codeword \mathbf{x} using the enumeration algorithm. During transmission of \mathbf{x} a single error is made, i.e. we receive \mathbf{x}' , $d_H(\mathbf{x}, \mathbf{x}') = 1$, where $d_H(\mathbf{x}, \mathbf{y})$ denotes Hamming distance between \mathbf{x} and \mathbf{y} . Translation using (6.6) will result in the word $\mathbf{b}' = i_S(\mathbf{x}') \neq \mathbf{b} = i_S(\mathbf{x})$, where \mathbf{b}' and \mathbf{b} are the binary representations of $i_S(\mathbf{x}')$ and $i_S(\mathbf{x})$, respectively. In particular we are interested in $d_H(\mathbf{b}, \mathbf{b}')$ and the error burst length distribution. The *error burst length*, b , is defined by $b = n_{\max} - n_{\min} + 1$, where n_{\min} and n_{\max} denote the smallest and largest positions where \mathbf{b}' and \mathbf{b} differ.

If an error is made at position k of the codeword, then the decoder will invoke (6.6) or Proposition 6.3 and form the inner product

$$\begin{aligned} i_S(\mathbf{x}') &= \sum_{j=1}^n x_j \hat{N}(n-j) + a \hat{N}(n-k) \\ &= i_S(\mathbf{x}) + a \hat{N}(n-k), \\ &= \mathbf{b} + a \hat{N}(n-k), \end{aligned} \tag{6.28}$$

where $a = 1$ if $x_k = 0$ or $a = -1$ if $x_k = 1$. All additions (or subtractions) are in binary notation. Clearly, severe error propagation can only occur if the binary addition (or subtraction) of $\mathbf{b} = i_S(\mathbf{x})$ and $\hat{N}(n-k)$ results in a long carry.

An analysis of the error statistics can be made if we make some assumptions. It is assumed that the source word \mathbf{b} is a random binary vector of doubly infinite length. Secondly, the mantissa of a weight $\hat{N}(n-k)$ is the binary p -vector $\mathbf{y} = (y_{p-1} \dots, y_0)$. By definition $y_{p-1} = 1$. The remaining $(p-1)$ elements are assumed to be random. If the above assumptions hold, the next theorem provides the error burst length distribution.

Theorem 6.1 *The error burst length distribution, $q(b)$, is given by*

$$q(b) = \begin{cases} \left(\frac{1}{2}\right)^p & , \quad b = 1, \\ \left(\frac{1}{2}\right)^{p-b+2} & , \quad 2 \leq b \leq p, \\ \left(\frac{1}{2}\right)^{b-p+1} & , \quad b > p, \end{cases} \tag{6.29}$$

and $q(b) = 0$ for $b \leq 0$.

Proof: See [163].

From Theorem 6.1 it is clear that the most likely burst has a length of p or $p + 1$ bits with probability $q(p) \approx q(p + 1) \approx 1/4$. Error bursts longer or smaller than p or $p + 1$ have an exponentially decaying probability.

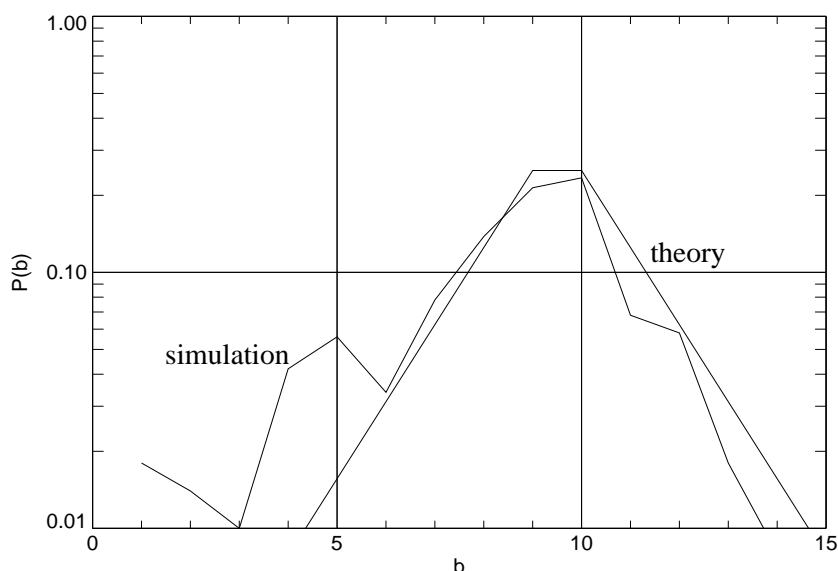


Figure 6.3: Comparison of error burst distribution for $d = 2$ and $p = 9$. After Immink & Janssen [163].

Figure 6.3 compares the results of computations of a typical example of computer simulations and computations using (6.29). The figure shows reasonable agreement of the theoretical distribution (6.29) with the distribution obtained by simulation. Thus we can control error propagation by a proper choice of p . The choice of the parameter p has an effect on the maximum achievable code rate (see (6.15)). With (6.29) and (6.15) a trade-off has to be made between, on the one hand, the error propagation effects, which has a bearing on the required capability of the error control code, and, on the other hand, the rate of the constrained code. This is a very subtle trade-off requiring a detailed specification of the various coding layers, and has therefore not been pursued.

6.5 Alternatives to standard codes

In order to limit error propagation inherent in long codes, various coding configurations have been developed, which do not follow the standard 'Figure 1' coding format as described in the Introduction to this book. In 'Figure

1', see Figure 1.1, page 3, source data are translated in two successive steps: (a) error-correction code (ECC) and (b) recording (or channel) code. The ECC encoder generates parity check symbols. The source data plus parity check symbols are translated into a constrained output sequence by the recording code, and the constrained output sequence, in turn, is stored on the storage medium.

Bliss [37], Fitinghof & Mansuripur [83], McMahan *et al* [243], Fan & Calderbank [79], and Immink [154] have proposed to revert the 'Figure 1' order of application of the error-correction code and the channel code. In the sequel, it is assumed that a long code is used in conjunction with a byte-oriented Reed-Solomon (RS) ECC. Reggiani & Tartara [288] have presented a study on a decoding procedure of the 'reverted' coding scheme. A block diagram of Bliss' scheme is shown in Figure 6.4.

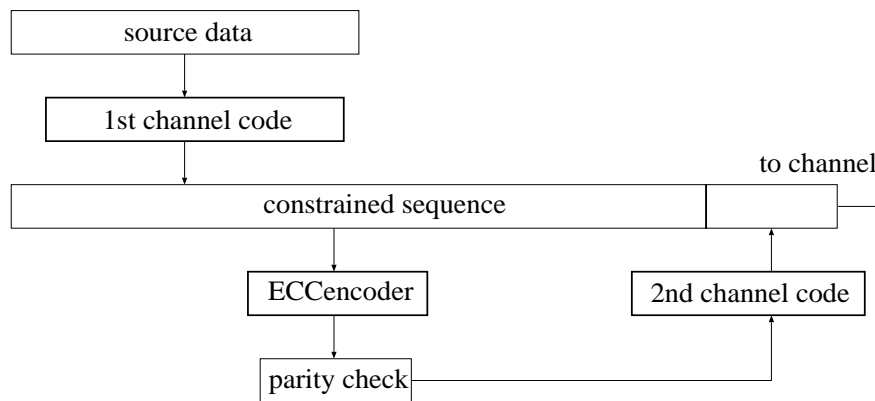


Figure 6.4: Code configuration of post-modulation or 'Bliss' scheme'.

In Bliss' coding format, a block of user data is translated, using a first constrained code, into a (long) constrained codeword. Then the constrained codeword is grouped into bytes and treated as input data of the RS error correcting code in the usual way. Parity bytes are generated under the rules of the RS code. The parity bytes thus generated do not, in general, obey the prescribed constraints and they are translated with the aid of a second constrained code. Provisions have to be made for concatenating the various segments. Decoding is straightforward. We start by decoding the parity information using the second channel code decoder. After a correction of possible errors in the constrained sequence, the corrected sequence is forwarded to the first constrained decoder, which, in turn, delivers the source sequence. Note that, unless the ECC decoder is overloaded, the input of the first constrained decoder is free of errors, and as a consequence there is no error propagation at this point. It is of paramount importance that the error propagation of the second channel code is limited to a few bits, preferably

to one or two bytes in a byte-oriented system. In principle, codes developed with the state-splitting algorithm or other constructions presented in Chapter 7 can be used, but usually additional buffering is required for cascading the various segments.

The efficiency of the second channel code is, as it is assumed that its length is smaller than that of the first channel code, (usually) lower than that of the first code. However, as the number of parity bytes is normally a small fraction of the number of input bytes, the efficiency of the second code has a relatively small bearing on the overall efficiency.

6.5.1 Burst error correction

In the above ‘Bliss’ scheme’, the constrained sequence is the input of the ECC code. Clearly, the constrained sequence is a factor of $1/R_1$ longer than the source data, where R_1 is the rate of the 1st channel encoder. Since the ECC operates on channel bytes, the corresponding number of user bytes it can correct is diminished by a factor of R_1 . For recording systems this implies that the burst error correction capability measured in geometrical units, e.g. meters, is reduced by the same factor R_1 . The length of the constrained sequence instead of the user sequence must be smaller than the maximum imposed by the RS code at hand.

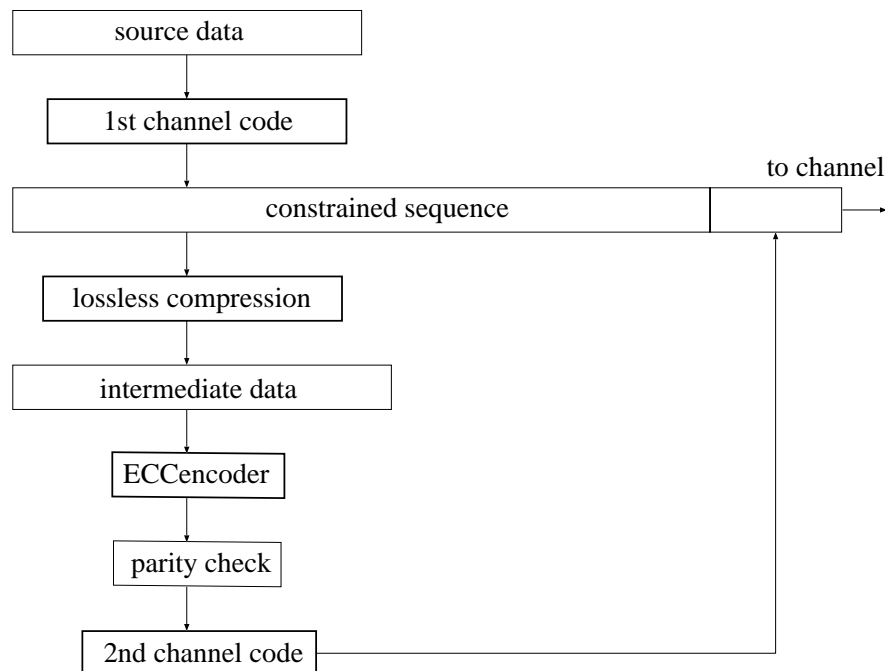


Figure 6.5: Encoder configuration of the new coding scheme.

The above drawbacks of Bliss' scheme are so severe, in particular when R_1 is low, that in spite of its efficiency benefits, it is of limited practical usefulness in recording systems, where correction of burst errors is a major requirement. These difficulties can be solved by reconfiguring the codes and defining a third intermediate coding layer. The new encoding format is shown in Figure 6.5. Essentially, the constrained sequence is compressed into a third intermediate sequence before it is forwarded to the ECC encoder. The constrained sequence is partitioned into blocks of q bits. The block size q is chosen so that the number of distinct constrained sequences of length q is not larger than N , the field size of the symbol error correcting ECC. Then it is possible to define a one-to-one mapping between the q -tuples and the ECC symbols.

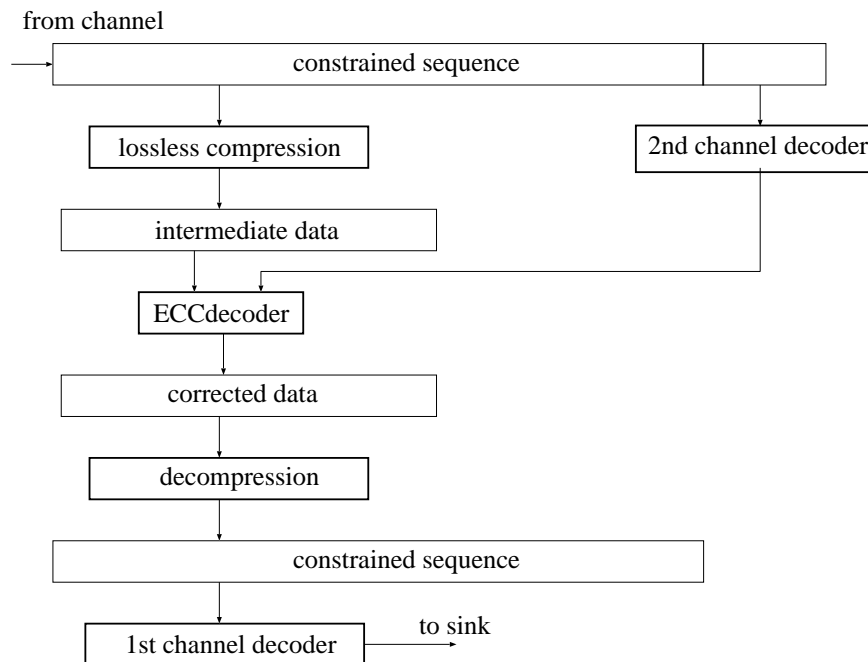


Figure 6.6: Decoder configuration of the new coding scheme.

Using a small look-up table \mathcal{L} , or enumeration, the q -tuples are uniquely translated into an intermediate sequence of bytes. The number of bytes so generated is slightly larger than the original number of source bytes. The intermediate sequence, in turn, is used as the input of the ECC encoder and the parity check bytes are generated as usual. It should be appreciated (see Figure 6.5) that the intermediate sequence is not transmitted. As in Bliss' scheme, the parity check bytes are encoded by a 2nd constrained code. The

cascaded sequence, i.e. the constrained sequence followed by the constrained parity bytes, is eventually transmitted. Decoding is straightforward, as can be seen in Figure 6.6, which shows a diagram of the set-up. First, the parity symbols are found using the second channel decoder. Then, after the ECC decoding operation, the corrected bytes are translated into the constrained sequence using the inverse of the look-up \mathcal{L} . The corrected constrained sequence is decoded by the 1st channel decoder.

Though the loss in burst error correcting capacity of the new construction is much less than that of Bliss' scheme, Figure 6.4, there is still some loss with respect to the conventional 'Figure 1' construction. The loss is simply the ratio of the lengths of the compressed sequence and the source sequence. It can easily be verified that $q = 11$ is the largest word length for which the number of $d = 1, k = 12$ sequences of a length q is less or equal 256. Thus the compression scheme translates the 371-bit dk sequence into $\lceil 371/11 \rceil = 34$ bytes. The expansion of the ECC input sequence, with respect to the conventional "Figure 1" configuration, is thus $34/32 = 1.06$.

Using Construction 3, it is possible to design a rate $256/466$, $(2,15)$ code, $p = 11$, whose efficiency is $R/C(2, 15) = 0.9986$. The rate of the new code is almost 10% larger than that of the traditional rate $1/2$, $(2,7)$ code. A possible lossless compression scheme has an input word length $q = 13$. Then the 466-bit input sequence is translated into $\lceil 466/13 \rceil = 36$ bytes, which results in an expansion factor of $36/32 = 1.125$. Alternatively, we may opt for a more complex compression scheme, which translates $q = 28$ bits into 2 bytes. This results in a smaller expansion factor, 1.06, but the increased error propagation (a single channel bit error may result in two decoded byte errors) puts an extra load on the ECC. The trading of the various parameters is a subtle matter requiring more study.

Fan, Marcus & Roth [80] showed that *sliding-block* compression schemes can be constructed. They disclosed, among others, a compression scheme of compression ratio $4/7$ that can be used to compress $(d = 2, k = \infty)$ sequences. The slightly increased error propagation of sliding-block compression seems negligible.

6.6 Appendix: Asymptotics

Assume a generating function, $T(x)$, can be written as the quotient of two polynomials, or

$$T(x) = \frac{q(x)}{p(x)} = \sum c_n x^n,$$

and that the polynomials $p(x)$ and $q(x)$ do not have factors in common. Let $p(x)$ have distinct roots and let the degree of $p(x)$ be m , then we can write

down $p(x)$ as a product of factors, namely

$$p(x) = K \prod_{i=1}^m (1 - \lambda_i x). \quad (6.30)$$

If the degree of $q(x)$ is smaller than m , we may write

$$T(x) = \frac{q(x)}{p(x)} = \sum_{i=1}^m \frac{A_i}{(1 - \lambda_i x)}. \quad (6.31)$$

By observing that $1/(1 - \lambda_i x)$ can be expressed as a geometric series we can write $T(x)$ as

$$T(x) = A_1(1 + \lambda_1 x + \lambda_1^2 x^2 + \dots) + A_2(1 + \lambda_2 x + \lambda_2^2 x^2 + \dots) + \dots$$

We easily find that the coefficient c_n equals

$$c_n = A_1 \lambda_1^n + \dots + A_m \lambda_m^n.$$

Let for clerical convenience $\lambda = \lambda_1$ then

$$c_n = A_1 \lambda^n \left(1 + \frac{A_2}{A_1} \left(\frac{\lambda_2}{\lambda}\right)^n + \dots + \frac{A_m}{A_1} \left(\frac{\lambda_m}{\lambda}\right)^n\right).$$

Assume λ is the largest of all λ_i , then we get the asymptotic relationship

$$c_n \approx A_1 \lambda^n.$$

An estimate of A_1 is given below. If we multiply (6.31) with $(1 - \lambda x)$ and set $x = 1/\lambda$, the only surviving term is the $A = A_1$ term. Therefore we have

$$A = \lim_{x \rightarrow 1/\lambda} T(x)(1 - \lambda x) = \lim_{x \rightarrow 1/\lambda} \frac{q(x)}{p(x)}(1 - \lambda x).$$

Now note that

$$\frac{p(x)}{1 - \lambda x} = \frac{p(x) - p(1/\lambda)}{1 - \lambda x} = -\frac{1}{\lambda} \frac{p(x) - p(1/\lambda)}{x - 1/\lambda}$$

Now let $x \rightarrow 1/\lambda$ then we find

$$\frac{p(x)}{1 - \lambda x} = -\frac{p'(1/\lambda)}{\lambda}.$$

As $p(x)$ has a single zero at $x = 1/\lambda$, we conclude $p'(1/\lambda) \neq 0$. As $q(x) \rightarrow q(1/\lambda)$ if $x \rightarrow 1/\lambda$, we find

$$\lim_{x \rightarrow 1/\lambda} \frac{q(x)}{p(x)}(1 - \lambda x) = -\lambda \frac{q(1/\lambda)}{p'(1/\lambda)}.$$

The constant A is

$$A = -\lambda \frac{q(1/\lambda)}{p'(1/\lambda)}, \quad (6.32)$$

so that

$$c_n \approx -\frac{q(1/\lambda)}{p'(1/\lambda)} \lambda^{n+1}. \quad (6.33)$$

Chapter 7

Sliding-Block Codes

7.1 Introduction

As we have learnt in the preceding chapter, attempts to increase the efficiency of block codes result in increased codeword length, and thus in rapidly mounting coder and decoder complexity. Codes that can be decoded with sliding-block decoders have, as we will demonstrate a high efficiency, small hardware, and not too many drawbacks. As will be discussed in Section 7.2, a sliding-block decoder observes the n -bit codeword plus r preceding n -bit symbols plus q later n -bit symbols. The decoder comprises a $(r+1+q)$ -stage shift register and decoder logic. The quantity $w = r + 1 + q$ is often called the *window size* of the decoder. A drawback of codes decoded by a sliding-block decoder is *error propagation* as the decoding operation depends on $r + q + 1$ consecutive codewords. In practice, the increased efficiency and reduced hardware of sliding-block decoder outweigh the extra load on the error correction unit. There are various coding formats and design methods for constructing such codes.

Variable-length codes, or, in short, VL codes, are excellent examples of a type of codes that can –if proper measures have been taken– be decoded with a sliding-block decoder. Such codes often permit a marked reduction in coder and decoder complexity relative to a fixed-length code of like rate and sequence properties. VL codes have the disadvantage that blocks of information will be translated into sequences of –indeed– variable length. In *synchronous VL codes*, on the other hand, the source words and the codewords are of variable length, but the quotient of the lengths of a source word and its associated codeword is fixed, so that blocks of information will be translated into blocks of encoded data of fixed length. The basis of VL synchronous codes was laid by Franaszek [96]. VL synchronous codes will be discussed in Section 7.3. Thereafter, in Section 7.4, we will discuss *look-ahead encoding techniques*, where the generated codeword is a function of the source word, the encoder state, and a limited number of upcoming

source words. In Section 7.5, we will discuss the sliding-block code algorithm, usually called *ACH algorithm* after the creators of the algorithm, Adler, Coppersmith & Hassner [2]. The design algorithm guarantees to find constrained codes, whose rate, m/n , is less or equal than channel capacity. The code found using the ACH algorithm has the virtue that it can be decoded with a sliding-block decoder of finite size (and thus limited error propagation). The ACH algorithm is a great step in the history of constrained code design. In the next section, we will start with a description and properties of the sliding-block decoder.

7.2 Description of a sliding-block decoder

A decoder observes the encoded sequence, usually after it has been transmitted or stored, and translates it into, hopefully, the original source sequence. Firstly, the received sequence is partitioned into n -bit codewords. Traditionally the partitioning of the received sequence, the *synchronization arrangement*, is said not to be part of the decoder's function, and we will tacitly assume that such a partitioning has taken place elsewhere. A state-dependent decoder translates a string of n -bit codewords into a string of m -bit source words. The encoder state is not, as such, transmitted by the sender, and therefore the translation depends on an *estimate* of the encoder state. As error correction usually takes place after this stage of (constrained) decoding, the decoder has to deal with possible channel errors that were made during transmission or storage. A state-dependent decoder could lose track of the encoder state sequence with the result that the decoder will generate an unbounded string of errors. Clearly, a state-dependent decoder is prone to severe error propagation as there is no guarantee that the decoder will ever again find the correct encoder's state.

Table 7.1: Codebook of three-state $R = 1/2$ code.

β	$h, g(\sigma_1, \beta)$	$h, g(\sigma_2, \beta)$	$h, g(\sigma_3, \beta)$
00	0110, σ_1	0110, σ_2	0110, σ_3
01	1001, σ_1	1001, σ_2	1001, σ_3
10	1010, σ_2	1010, σ_3	0011, σ_1
11	1100, σ_3	0101, σ_1	0101, σ_2

A *practical* decoder should have the property that any finite number of input errors gives rise to a finite number of output errors. A simple example of a code that allows such a decoder is a code that solely requires an n -bit codeword as an input to regenerate the m -bit source word. The state-dependent

translation of source words $\{\beta_u\}$ into codewords $\{\chi_{iu}\}$ with the output function $h(\sigma_i, \beta_u)$ must preferably have an unambiguous inverse mapping $h^{-1}(\chi_{iu}) = \beta_u$ without reference to the encoder state σ_i . Let us take a look at the encoder table in Table 7.1, which describes a state-dependent encodable encoder. It can easily be verified that this state-dependent code can be decoded state-independently using a block decoder with the Boolean table presented in Table 7.2. Observation of single 4-bit code words makes it possible to uniquely retrieve the source message without referring to the encoder state, or an estimate of the encoder state.

Table 7.2: Decoding table of three-state $R = 1/2$, code.

χ	β
0110	00
1001	01
1010, 0011	10
1100, 0101	11

A more general type of decoder requires besides the n -bit codeword, r past n -bit symbols plus q future n -bit symbols for uniquely translating the n -bit codeword into the source word. Such a decoder is called a *sliding-block decoder*.

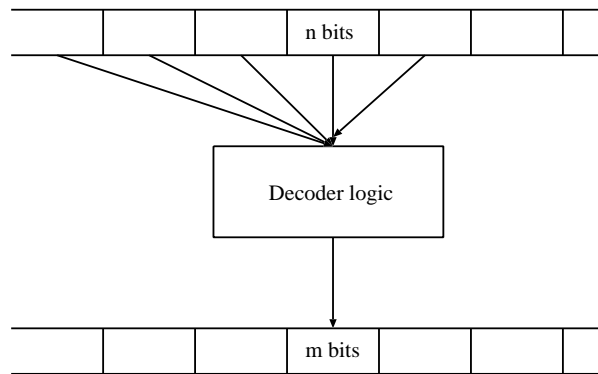


Figure 7.1: Schematic diagram of a sliding-block decoder. A source word is retrieved after observing the current, r past, and q future codewords.

Figure 7.1 shows a schematic diagram of a sliding-block decoder. Essentially the decoder comprises a $w = (r + 1 + q)$ -stage shift register and decoder logic. We will refer to the quantity w as the *window size* of the decoder. A drawback of codes that are decoded by a sliding-block decoder is *error propagation* as the decoding operation depends on $r + q + 1$ consecutive

codewords. Clearly, single errors at the input to a sliding-block decoder have a finite error propagation as the erroneous words will be shifted out after $(r + 1 + q)$ clock periods. The next example shows a typical instance of a code that requires a sliding-block decoder of window size $w = 2$.

Example 7.1 Consider the rate $1/2$, code with the codebook shown in Table 7.3. The code can be implemented with a 2-state encoder.

Table 7.3: Encoding table of two-state $R = 1/2$, code.

β	$h, g(\sigma_1, \beta)$	$h, g(\sigma_2, \beta)$
0	00, σ_1	01, σ_1
1	00, σ_2	10, σ_1

It can easily be verified that the output sequence generated by the encoder has at least two 'zeros' between consecutive 'ones'. Note that in state σ_1 the source words '0' and '1' are both represented by '00', and that, thus, the observation of a single 2-bit codeword is not sufficient to retrieve the source word. Perusal of the encoding table shows that the codeword '00' when it represents the source word '0' is followed by the codewords belonging to state σ_1 , i.e. codeword '00'. The codeword '00' representing the source word '1' is always followed by codewords of state σ_2 , namely '10' or '01'. Therefore the decoding ambiguity can be resolved by looking ahead one 2-bit codeword. Thus in the above terminology we have $q = 1$ and $r = 0$. The codewords '10' and '01' can be decoded without look-ahead.

Table 7.4: Decoding table of two-state $R = 1/2$, code.

χ_t	χ_{t+1}	β_t
00	01	1
00	10	1
00	00	0
10	don't care	1
01	don't care	0

The decoding table is shown in Table 7.4, where $\chi_t, \chi_{t+1}, \dots$ denotes the string of codewords received at instants $t, t + 1$ etc.

7.3 Variable-length (VL) codes

Variable-length (VL) (sometimes called variable-rate) codes can easily be designed, as can be seen from the example given in Table 7.5, which constitutes a $(1, \infty)$ RLL code. Similar codes can be written down for other

runlength constraints as well. Lossless source codes can be used as variable-length RLL codes as was demonstrated by Kerpez [201]. If it is assumed that the input data of the code listed in Table 7.5 is 'random', i.e. the source 1s and 0s are independent and occur at probability $1/2$, then the average rate of the above code is $2/3$, which is only a few percent below channel capacity. The above scheme can be seen as a bit stuffing method, where a '0' is inserted, 'stuffed', after every source '1'. Though this code is very simple, there are, unfortunately, some serious practical drawbacks. In real life, input data is seldom random, and as a result, for example, if the input data would consist of the all '1's string the 'rate' of the code would be $1/2$. In other words, the code is very sensitive for worst case inputs. In an effort to transform the incoming data stream into a form that is amenable to variable-length encoding the incoming data stream is first randomized using, for example, a pseudo-random binary sequence, which is 'exclusive-OR'ed with the incoming data stream to give the resulting output a random or pseudo-random character.

Table 7.5: Variable-length $(1, \infty)$ code.

<i>Data</i>		<i>Code</i>
0	$\leftarrow \rightarrow$	0
1	$\leftarrow \rightarrow$	10

The randomization procedure is usually executed by a scrambler or even multiple scramblers, see Section 10.2.1 or [112, 217]. If it is assumed that the scrambled data is sufficiently random, then the length of the output string is a random variable. This makes it difficult to practically realize a system that works with fixed-length blocks required in hard-disk drives or optical storage systems. It has also been found that in certain instances, when the incoming data pattern has a characteristic that correlates with the pseudo-random binary sequence (i.e. the scrambler polynomial) in such a way as to produce undesirable encoded characteristics, the randomization using that polynomial does not act to prevent the length of the VL output sequence being statistically too long. A data stream having this characteristic is typically referred to as a *degenerate pattern*. Simply the possibility of occurrence of such a degenerate pattern has generally been considered as an impediment to the use of VL coding systems in the data storage environment. The usage of a high-rate variable length $(0, k)$ code, where multiple scramblers are used to circumvent the effects of degenerate patterns, has been reported in the literature [114, 250]. Though it is not impossible to accommodate output blocks of variable length, it is not very difficult, as will be shown shortly, to design very efficient codes that do generate fixed-length

output blocks. So one could well argue that there is no practical need for such variable-length codes as shown in Table 7.5. VL codes that generate output blocks of fixed length are termed *synchronous variable length (VL) codes*.

The structure of VL synchronous codes is very similar to that of fixed-length codes. Various special features, however, arise from the presence of words of different lengths. The requirement of synchronous transmission, coupled with the assumption that each word carries an integer number of information bits, implies that the codeword lengths are integer multiples of a basic word length n , where n is the smallest integer for which the bit per symbol ratio m/n is that of two integers. One might argue that the problem, as discussed above, of generating sequences of variable length that do not fit into a fixed length frame format, is not completely solved by the adoption of synchronous VL codes. That is true, but in most practical codes the maximum codeword is small, and usually a special structure can be designed for a proper termination of the sequence. For an example see [24].

In the next section, we will discuss this important type of codes.

7.3.1 Synchronous variable-length RLL codes

The existence of a VL synchronous code having certain specifications can be established with a modification of Franaszek's recursive procedure, see Section 5.3.1, page 100. To apply Franaszek's modified recursive procedure, a basic codeword length n and source word length m are chosen. Codewords may be of length jn , $j = 1, 2, \dots, M$, where Mn is the maximum codeword length. The routine involves operations on powers of the adjacency matrix, which is reminiscent of Franaszek's recursive procedure for designing block codes. As in the original recursive procedure, the objective is to find a set of principal states. If the recursive procedure is successful, the outcome is a set of principal states, denoted by Σ_p , from which coding may be performed.

Decoding of the VL codes to be discussed can be effected without explicitly knowing where the blocks of variable length start or end, that is, the codes are *self-punctuating* (the n -bit synchronization is supposed to be maintained by a separate device). The codes are self-punctuating, because they satisfy the *prefix condition*. A VL synchronous code is a set of $S = \{\mathbf{c}_0, \dots, \mathbf{c}_{M-1}\}$ of M strings. If a codeword $\mathbf{c}_u \in S$ is not the beginning of $\mathbf{c}_v \in S$ for any $u \neq v$ and for all u , then the code is called a *prefix code*.

7.3.2 Examples of synchronous VL codes

The above recursive algorithm can be programmed quite easily. There is, however, a method that gives more insight. Assume the design of a rate

$m/n, (d, \infty)$ code. Write down all words of length n that end with d '0's. Then write down all words of length $2n$ that end with d '0's and do not have said n -bit words as a prefix. Repeat this procedure for codewords of length $jn, j = 3, \dots, M$, until sufficient words have been found. A $(d, k < \infty)$ code can be constructed as follows.

Write down all $(dklr)$ sequences, $l+r = k$ of length $n, 2n, 3n, \dots$, where we prohibit those words that have a prefix equal to a shorter word that was already written down; the all '0's word, '0...0', is excluded. The value of $l < k$ is chosen by trial and error. Three excellent representatives of prefix VL synchronous codes, to be discussed in the following case studies, are due to Franaszek [90].

Rate 1/2, $(1, \infty)$ code

We choose the same runlength parameters as in Example 5.2, page 102, namely $d = 2, k = \infty$. As shown in Example 5.2, the minimum codeword length of a rate 1/2 block-decodable code is $n = 14$. This code needs a look-up table of 128 entries for encoding and decoding. With the above VL code algorithm, a code comprising only three words can be easily constructed with $m = 1$ and $n = 2$, as shown in Table 7.6.

Table 7.6: Variable-length synchronous $R = 1/2, (2, \infty)$ code.

<i>Data</i>		<i>Code</i>
0	← →	00
10	← →	0100
11	← →	1000

If the input symbol is '0', '00' would be transmitted. Otherwise, the encoder would transmit either '0100' or '1000' depending on whether the next symbol is a '0' or a '1', respectively. By inspection it is clear that the three variable-length codewords can be cascaded without violating the $d = 2$ channel constraint. Note that the code satisfies the prefix condition.

Table 7.7: Finite-state machine description of simple rate 1/2, $(2, \infty)$ variable-length synchronous code.

β_i	$h(\sigma_1, \beta_i)$	$g(\sigma_1, \beta_i)$	$h(\sigma_2, \beta_i)$	$g(\sigma_2, \beta_i)$
0	00	σ_1	01	σ_1
1	00	σ_2	10	σ_1

The encoding scheme of VL synchronous codes may be readily implemented in practice. Alternatively, the above scheme can be described in terms of a 2-state finite-machine encoder, whose characteristics are shown in Table 7.7.

The above elementary example illustrates quite well the great advantage of the VL synchronous coding approach in terms of hardware, and it actually shows how the fixed-length block code with a 128-word dictionary (see Example 5.2, page 102) may be replaced by one with only three words.

Rate 1/2, (2,7) code

The VL code pointed out in the previous example can be slightly modified to incorporate a maximum runlength constraint k . Table 7.8 discloses the code table of the rate 1/2, (2,7) code, which constituted the bedrock of high-performance hard-disk drives [145].

Table 7.8: Variable-length synchronous rate 1/2, (2,7) code.

<i>Data</i>		<i>Code</i>
10	← →	1000
11	← →	0100
011	← →	000100
010	← →	001000
000	← →	100100
0011	← →	00100100
0010	← →	00001000

The codebook comprises seven codewords that start with at most $l = 4$ and end with at most $r = 3$ 'zero's. The encoding of the incoming data is accomplished by dividing the source sequence into 2-, 3-, and 4-bit partitions to match the entries in the code table and then mapping them into the corresponding channel representations. The next example describes how the codebook is to be used. Let the source sequence be 010111010, then after the appropriate parsing, we obtain

$$in : 010\ 11\ 10\ 10\ \dots,$$

which, using Table 7.8, is transformed into the corresponding output sequence

$$out : 001000\ 0100\ 1000\ 1000\ \dots$$

The companion Table 7.9 shows the same codewords and a permutation of the codeword assignments (there are 24 permutations of the above correspondences). It is worth pointing out here that the assignment rules found

by Eggenberger & Hodges [72], which at first sight seem (again) quite arbitrary, are the outcome of a judicious choice, which will become clear in the following.

The code satisfies the prefix condition and is therefore self-punctuating. In the case of Table 7.9, decoding of the received message can be achieved with a sliding block decoder of length four (2-bit) words. Error propagation is limited: any error in a received bit may entail a decoding error in up to two subsequent data bits, the current data bit and up to one preceding data bit. Thus, no error in a received bit is propagated beyond at maximum four decoded data bits. Detailed computations of the effects of shift error propagation of the (2,7) code have been conducted by Howe and Hilden [144].

Table 7.9: Variable-length synchronous rate 1/2, (2,7) code.

<i>Data</i>		<i>Code</i>
10	← →	0100
11	← →	1000
011	← →	001000
010	← →	100100
000	← →	000100
0011	← →	00001000
0010	← →	00100100

The code listed in Table 7.8, originally presented by Franaszek [90], has the drawback that it needs a shift register of length six 2-bit words, which increases error propagation to at most six decoded symbols. This instance demonstrates that the allocation of codewords in a VL code may have a crucial effect on the error propagation characteristic of the code. How the assignments should be chosen in order to minimize error propagation is up till now an unsolved problem.

Perusal of Table 5.6, page 104, reveals that the shortest fixed-length block code that generates a rate 1/2, (2,7) code has codeword length 34. Evidently, the VL synchronous code is much more attractive with respect to hardware requirements.

Rate 2/3, (1, k) codes

After the previous examples of rate 1/2, $d = 2$ codes, the design of a rate 2/3, (1, ∞) code is elementary. Write down all ($d = 1$) constrained words of length three that end with a '0', i.e. '000', '010', and '100'. Then write down all words of length six that end with a '0' and do not have said 3-bit

words as a prefix. There are four codewords that satisfy these conditions, namely '001000', '001010', 101000, and '101010'. We tag source words, and the result, a code comprising seven codewords of length 3 and 6, is shown in Table 7.10.

Table 7.10: Variable-length synchronous $R = 2/3$, $(1, \infty)$ code.

<i>Data</i>		<i>Code</i>
00	← →	000
01	← →	010
10	← →	100
1100	← →	001000
1101	← →	001010
1110	← →	101000
1111	← →	101010

Table 7.11: Variable-length synchronous $R = 2/3$, $(1,8)$ code.

<i>Data</i>		<i>Code</i>
00	← →	010
01	← →	100
1000	← →	101000
1001	← →	101010
1010	← →	001000
1011	← →	000100
1100	← →	000010
1101	← →	001010
111000	← →	000001010
111001	← →	000001000
111010	← →	000101010
111011	← →	000101000
111100	← →	001001010
111101	← →	001001000
111110	← →	101001010
111111	← →	101001000

We will now construct a rate $2/3$, $(1,8)$ code. The design runs in a similar vein. The all '0's word cannot be used. Now we write down all $(dklr)$ sequences, $l + r = k$ of length 3, 6, ..., prohibiting those words that have a prefix equal to a shorter word that was already written down. The value of

$l < k$ is chosen by trail and error. In the rate $2/3$, $(1,8)$ code, words start with at most $l = 5$ '0's and end with at most $r = 3$ '0's. This example of a VL synchronous code was given by Franaszek [90]. The coding table, comprising 16 words of length 3, 6 and 9, is presented in Table 7.11. No attempts have been made to optimize the coding table for minimum error propagation or other virtues. Note that if we choose, for symmetry reasons, $l = r = 4$, a rate $2/3$, $(1,8)$ code can only be constructed if we allow words of length 12. A rate $2/3$, $(1,7)$ code, whose details are omitted, can be constructed by choosing $l = 4$ and $r = 3$. The code consists of 32 words of length 3, ..., 18. Decoding of the above VL prefix codes can be accomplished by a sliding block decoder. Table 7.12 presents the parameters of variable-length codes that have been published in the literature, see Gabor [106], Franaszek [89], Kobayashi [208], Horiguchi, and Morita [143]. A variable-length encoder can always be cast in the standard format of a finite-state machine encoder, which does not mean that a finite-state machine encoder is the preferred embodiment of an encoder.

Table 7.12: Parameters of published RLL codes.

d	k	m	n	M	R
0	2	4	5	1	$4/5$
0	3	9	10	1	$9/10$
1	3	1	2	1	$1/2$
1	7	2	3	2	$2/3$
2	7	1	2	4	$1/2$
3	7	2	5	8	$2/5$
4	14	4	11	3	$4/11$
5	17	1	3	6	$1/3$

A few remarks are in order regarding the usage of VL synchronous codes in the fixed-length prefix synchronized frame format. In magnetic or optical mass storage systems, the coded information is commonly grouped in large blocks, called *frames*. At the beginning of each frame we will find the synchronization pattern, which is used by the receiver to identify the frame boundaries. The problem is that the encoded data must fit into the interval provided, and that the VL synchronous encoder looks ahead a limited number of source words. This difficulty can be overcome by defining substitute encoder and decoder tables that are used just prior to the occurrence of a sync pattern.

7.4 Look-ahead encoding technique

Another class of design techniques documented in the literature [220, 172, 274, 62, 174, 91, 88] is called *future-dependent* or *look-ahead* coding. Notably Hollmann [139] presented major contributions to the art. A block code is said to be look-ahead if the encoding operation of a current block may depend on upcoming symbols. In a generalization of this type of codes, called *bounded-delay encodable codes* [92, 93, 94] codewords 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 efficient RLL codes.

The code design is guided by the approximate eigenvector inequality. The approximate eigenvector also plays a key role in another design method, the ACH algorithm, see Section 7.5. Let the code rate be $m/n < C(d, k)$, where m and n , $m < n$, are positive integers, and let D denote the $N \times N$ adjacency matrix. An approximate eigenvector \mathbf{v} is a non-negative integer vector, in this context not necessarily two-valued, satisfying

$$D^n \mathbf{v} \geq 2^m \mathbf{v}. \quad (7.1)$$

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. If λ is the largest positive eigenvalue of the connection matrix D then by the Perron-Frobenius theory, there exists a vector \mathbf{v} whose entries v_i are integers satisfying (7.1), where $n/m \leq \log_2 \lambda$. The following algorithm, taken from Adler *et al.* [2], is an approach to finding such a vector.

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

$$v_i^{(u+1)} \equiv \min \left(v_i^{(u)}, \lfloor (2^{-m} \sum_{j=1}^N [D]_{ij}^n v_j^{(u)}) \rfloor \right). \quad (7.2)$$

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) $\mathbf{v} > \mathbf{0}$ and (b) $\mathbf{v} = \mathbf{0}$. Case (a) means that we have found an approximate eigenvector, and in case (b) there is no solution, so we increase L and start from the top again. There may be multiple solutions for the vector \mathbf{v} . Hollmann [140] showed that the choice of \mathbf{v} 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. After finding an approximate eigenvector by invoking (7.2), it is sometimes possible to find a better eigenvector by

a systematic trial and error method. If the components of \mathbf{v} are all zeros and ones, no look-ahead is required, and we may immediately construct the encoder as outlined in Section 5.3, page 99. In Section 5.7, page 122, various worked examples of look-ahead block-decodable codes can be found. An example of a code design based on the look-ahead method is the rate $2/3$, $(1,7)$ code to be detailed in the next section.

7.4.1 Rate $2/3$, $(1,7)$ code

Jacoby & Kost [174, 61] described a rate $2/3$, $(1,7)$ code with full-word look-ahead used in a particular magnetic disc drive. A similar code was found by Adler, Hassner & Moussouris [3] using the ACH algorithm (see Section 7.5, page 172). To better understand the $2/3$ -rate look-ahead code, we commence with the basic encoding table, presented in Table 7.13.

Table 7.13: Basic coding table $(1,7)$ code.

<i>Data</i>	<i>Code</i>
00	101
01	100
10	001
11	010

The $2/3$ -rate code is quite similar to a fixed-length block code, where data words of 2 bits are converted into codewords of 3 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. 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 'one's. The process of substitutions in these four combinations is revealed in Table 7.14. Error propagation during decoding is limited to five consecutive data bits. The encoding rules of Tables 7.13 and 7.14 can be cast into finite-state machine having five states [110]. Weathers, Swanson & Wolf [336, 337] presented a rate $2/3$, $(1,7)$ code that can be encoded with a 4-state encoder. The encoder was found with the ACH algorithm. Patel [274] also employed a look-ahead approach for the design of the Zero-Modulation code (see Chapter 11).

Table 7.14: Substituting coding table (1,7) code.

<i>Data</i>	<i>Code</i>
00.00	101.000
00.01	100.000
10.00	001.000
10.01	010.000

7.5 Sliding-block algorithm

In this section, we present a description of the sliding-block algorithm by Adler, Coppersmith & Hassner or ACH algorithm [2, 11]. The algorithm has been generalized by Marcus [231], Karabed [186] and Ashley & Marcus [15]. An excellent treatise of symbolic dynamics can be found in [224] by Lind & Marcus. For further reading see also the tutorial by Marcus, and Siegel & Wolf [235]. The sliding-block code algorithm guarantees the design of codes that can be decoded using a sliding-block decoder of finite length. For a given rate, $R = m/n \leq C$, a finite-state machine is derived during a finite rounds of modifications of the finite-state machine underlying the channel constraints. Eventually a finite-state machine is produced that has at least 2^m outgoing branches per state. The key idea in the construction method is that one modifies the finite-state machine describing the constrained channel by splitting and merging some of the channel states to obtain a new finite-state machine. During a state splitting operation new states are added to the machine, and it is not a prerequisite that the encoder states are a (sub)set of the channel states as it is, for example, in Franaszek's principal state method. The approximate eigenvector \mathbf{v} is used to guide the splitting process. Each state σ_i will be split into v_i encoder states. Some states can be merged so that the total number of encoder states is at most $\sum_i v_i$. The theory of the sliding-block code algorithm is not simple and it is full of subtlety. A simple example, taken from Siegel [300], may serve to illustrate the idea. A more detailed description will be given later.

Example 7.2 We construct a rate 2/3, (0,1) code. The connection matrices D and D^3 are

$$D = \begin{bmatrix} 1 & 1 \\ 1 & 0 \end{bmatrix}, \quad D^3 = \begin{bmatrix} 3 & 2 \\ 2 & 1 \end{bmatrix}. \quad (7.3)$$

The finite-state transition diagrams pertaining to D^3 is shown in Figure 7.2. An eigenvector inequality is given by

$$D^3 \mathbf{v} = \begin{bmatrix} 3 & 2 \\ 2 & 1 \end{bmatrix} \begin{bmatrix} 2 \\ 1 \end{bmatrix} \geq 2^2 \begin{bmatrix} 2 \\ 1 \end{bmatrix} = 2^2 \mathbf{v}. \quad (7.4)$$

The approximate eigenvector $\mathbf{v} = (2, 1)^T$ indicates that state σ_1 will be split into two states, while state σ_2 will not be split. In this example there will be three encoder states, because state σ_1 will split into two encoder states, while state σ_2 will remain unsplit. The two states into which state σ_1 is split are denoted by σ_{11} and σ_{12} . The outgoing edges of state σ_1 are partitioned into two groups which are assigned to the two 'offspring' states.

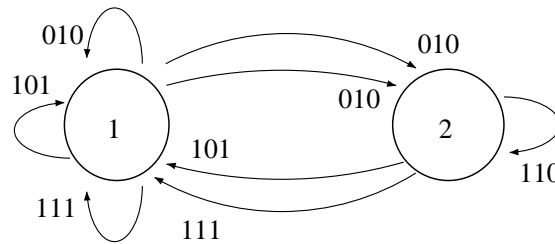


Figure 7.2: Finite-state transition diagram of third extension of (0,1) sequence. After Siegel 1985 [300].

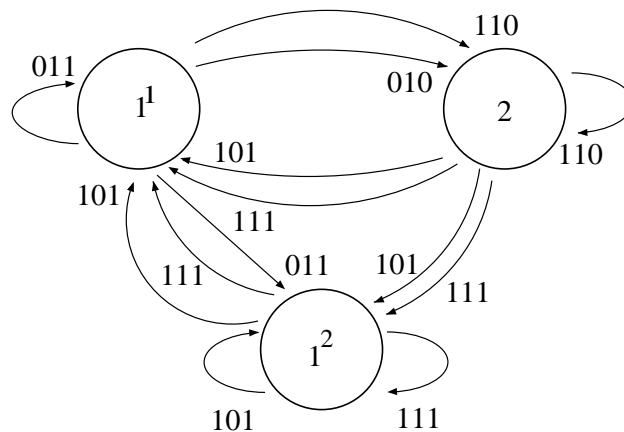


Figure 7.3: Split graph. After Siegel 1985 [300].

Decoding can be accomplished with a 6-bit shift register plus decoding logic, whose Boolean function is easily discerned from the encoder model, Figure 7.4. Error propagation is confined to at most four data bits.

All edges which entered state σ_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 Figure 7.3. It 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 Figure 7.4).

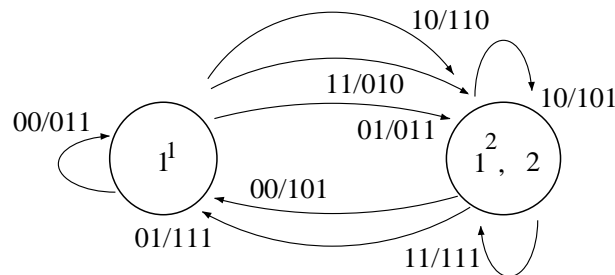


Figure 7.4: Finite-state machine representing a (0,1) code. After Siegel 1985 [300].

7.5.1 Higher-order edge graphs

Before we start with a description of the heart of the ACH algorithm, the state splitting, it is necessary to introduce a new concept: the higher-order edge graph. As usual the key of the code design is the state transition matrix D and its higher powers. The use of the ACH algorithm, as presented in Adler *et al.* [2], is often compounded by the fact that it requires an input matrix with 0/1 elements only. The given transition matrix, usually a power of D , may not have only 0/1 terms. Higher-order edge graphs, to be discussed in this section, provide a tool for constructing a source, described by a zero/one transition matrix, which is equivalent with the original source. The alternative representation of the original source is formed by making the allowed transitions, the edges, of the original source, called first order edge graph, into the states of a new equivalent source, called the second order edge graph. A new transition matrix is formed by specifying how the edges of the original source are connected. Likewise, edges of the second order edge graph play the role of states in the third order edge graph, etc. The transition matrix of the q th order edge graph is denoted by $D^{[q]}$. The output function pertaining to the q th order edge graph, denoted by $\zeta^{[q]}$, is implied by the output function and the structure of the original source.

Note that the number of states of the $(q + 1)$ th, $q \geq 1$, order edge graph equals the number of edges, that is, the number of elements equal to one of the q th order edge graph. The procedure is best explained by an example.

Example 7.3 Consider a rate $m/n = 2/3$, $(0,1)$ code. The transition matrix D and output vector ζ are

$$D = \begin{bmatrix} 1 & 1 \\ 1 & 0 \end{bmatrix}, \quad \zeta = \begin{bmatrix} 1 \\ 0 \end{bmatrix}. \quad (7.5)$$

The matrix

$$D^3 = \begin{bmatrix} 3 & 2 \\ 2 & 1 \end{bmatrix} \quad (7.6)$$

has terms > 1 so that, in principle, the ACH algorithm cannot be applied. The formation of higher-order edge graphs will solve this difficulty. The 2-state information source defined by (7.5) has three edges (the number of unity elements of D). The second order edge graph of this source is specified by the 3×3 matrix $D^{[2]}$ and the output function $\zeta^{[2]}$

$$\begin{array}{c} 11 \quad 12 \quad 21 \\ 11 \begin{bmatrix} 1 & 1 & 0 \\ 0 & 0 & 1 \\ 1 & 1 & 0 \end{bmatrix} \quad \zeta^{[2]} = \begin{array}{c} 11 \\ 12 \\ 21 \end{array} \begin{bmatrix} 1 \\ 0 \\ 1 \end{bmatrix}. \end{array} \quad (7.7)$$

The numbers above and to the left of the matrix correspond to the values of i and j for which $d_{ij} = 1$. A graphical representation of the second order edge graph is shown in Figure 7.5.

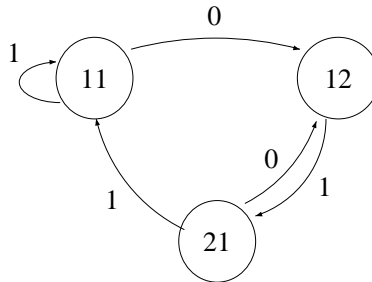


Figure 7.5: Graphical representation of the second order edge graph for a $(0,1)$ sequence.

The third power of $D^{[2]}$ is

$$D^{[2]^3} = \begin{array}{c} 11 \quad 12 \quad 21 \\ 11 \begin{bmatrix} 2 & 2 & 1 \\ 1 & 1 & 1 \\ 2 & 2 & 1 \end{bmatrix}, \end{array} \quad (7.8)$$

which contains elements > 1 , so that it is necessary to continue the process by forming the third order edge graph. We find

$$D^{[3]} = \begin{bmatrix} 1 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 1 \\ 1 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \end{bmatrix}, \quad \zeta^{[3]} = \begin{bmatrix} 1 \\ 0 \\ 1 \\ 1 \\ 0 \end{bmatrix} \quad (7.9)$$

and the third power of $D^{[3]}$:

$$D^{[3]^3} = \begin{bmatrix} 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 0 & 0 \\ 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 0 & 0 \end{bmatrix}, \quad \zeta^{[3]^3} = \begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 0 \\ 1 & 0 & 1 \\ 0 & 1 & 1 \\ 0 & 1 & 0 \end{bmatrix}. \quad (7.10)$$

This matrix is binary valued and can therefore be used as an input for the ACH algorithm.

We are now in position to describe the kernel of the ACH algorithm: the state splitting.

7.5.2 State splitting

Consider an N -state unifilar Moore-type Markov information source having a connection matrix T , $t_{ij} \in \{0, 1\}$, and output vector ζ . The matrix T does not necessarily describe dk constraints, and some of the above restrictions can be relaxed, see e.g. [186]). The key idea of the state splitting process is that such a source can be replaced by a new unifilar $(N + p)$ -state, $p \geq 1$, source with the property that sequences generated by the new source can be one-to-one transformed into a sequence generated by the old N -state source. The sequences generated by the two sources are essentially the same even if they look entirely different. Such a new source can be found with a process termed *state splitting*.

Let E_i denote the set of successors of the state $\sigma_i \in \Sigma$, and let $\{E_i^1, \dots, E_i^\alpha\}$, $\alpha = \alpha_i \geq 2$, be a disjoint partition of E_i , i.e. $E_i = E_i^1 \cup \dots \cup E_i^\alpha$. It is possible to construct a new $(N + \alpha - 1)$ -state unifilar Markov information source by replacing the state σ_i by α new states, $\sigma_{i1}, \dots, \sigma_{i\alpha}$. An edge that emanates from σ_i and terminates at $\sigma_j \in E_i^u$, $u = 1, \dots, \alpha$, is replaced by an edge that emanates from σ_{iu} and terminates at σ_j . Each edge that terminates in σ_i will terminate in the α new states unless $j = i$, that is, there is a loop at σ_i . In that case if $\sigma_i \in E_i^u$, the loop at σ_i is replaced by a loop at σ_{iu} and edges from σ_{iu} to σ_{iw} , $w \neq u$, respectively. The new $(N + \alpha - 1) \times (N + \alpha - 1)$ connection matrix is denoted by \hat{T} . The output of the α offspring states is equal to the output of the parent state, that is, $\zeta(\sigma_{i1}) = \dots = \zeta(\sigma_{i\alpha}) = \zeta(\sigma_i)$.

Example 7.4 Consider the 4×4 transition matrix T :

$$\begin{array}{c} 1 \quad 2 \quad 3 \quad 4 \\ \begin{array}{l} 1 \\ 2 \\ 3 \\ 4 \end{array} \begin{bmatrix} 1 & 0 & 1 & 0 \\ 1 & 1 & 0 & 1 \\ 1 & 1 & 0 & 0 \\ 0 & 1 & 0 & 0 \end{bmatrix} \end{array}.$$

Note, for example, that σ_2 has three successors, namely, σ_1 , σ_2 , and σ_4 . In order to illustrate the process, we split state σ_2 into $\alpha = 2$ offspring states σ_{2^1} and σ_{2^2} . If we choose $E_2^1 = \{\sigma_1, \sigma_4\}$ and $E_2^2 = \{\sigma_2\}$, we obtain the 5×5 matrix \hat{T} :

$$\begin{array}{c} 1 \quad 2^1 \quad 2^2 \quad 3 \quad 4 \\ \begin{array}{l} 1 \\ 2^1 \\ 2^2 \\ 3 \\ 4 \end{array} \begin{bmatrix} 1 & 0 & 0 & 1 & 0 \\ 1 & 0 & 0 & 0 & 1 \\ 0 & 1 & 1 & 0 & 0 \\ 1 & 1 & 1 & 0 & 0 \\ 0 & 1 & 1 & 0 & 0 \end{bmatrix} \end{array}$$

For convenience we demonstrated the splitting process of only one state. It is, however, possible to split more than one state per round. The above process of state splitting can be repeated a number of times. After each round of state splitting the number of states is increased by $\sum_i (\alpha_i - 1)$. At first sight, it seems to that we are making matters more complex than they are. The next paragraph will show the relevance of the state splitting.

Let the capacity of the unifilar source with transition matrix T be lower bounded by m , m an integer. Then Adler *et al.* [2] proved that there is an equivalent source having a connection matrix with row sums $\geq 2^m$. We give a brief outline of the algorithm. From the previous section, we know that there is an approximate eigenvector \mathbf{v} , $v_i \geq 0$, such that

$$T\mathbf{v} \geq 2^m \mathbf{v}. \quad (7.11)$$

Consider the set E_i of successors of σ_i . Let E_i^1, \dots, E_i^α be a disjoint partition of E_i that satisfies the conditions

$$\begin{aligned} \sum_{j \in E_i^u} v_j \bmod 2^m &= 0, \quad 1 \leq u \leq \alpha - 1, \\ v_i - 2^{-m} \sum_{u=1}^{\alpha-1} \sum_{j \in E_i^u} v_j &\geq 0, \end{aligned} \quad (7.12)$$

where, for convenience, the state σ_j is identified with the integer j . According to [2], there exists a partition such that condition (7.12) is satisfied. Once we have found such a state, say σ_i , we split σ_i into α offspring states $\sigma_{i^1}, \dots, \sigma_{i^\alpha}$. We form a new transition matrix \hat{T} and a new approximate eigenvector $\hat{\mathbf{v}}$ with components

$$(v_1, \dots, v_{i-1}, v_{i^1}, \dots, v_{i^\alpha}, v_{i+1}, \dots, v_N).$$

The weights of the α offspring states, $v_{i1}, \dots, v_{i\alpha}$, are calculated as follows:

$$\begin{aligned} v_{iu} &= 2^{-m} \sum_{j \in E_i^u} v_j, \quad 1 \leq u \leq \alpha - 1, \\ v_{i\alpha} &= v_i - \sum_{u=1}^{\alpha-1} v_{iu}. \end{aligned} \tag{7.13}$$

It is guaranteed by the splitting rules (7.12) that the weights of the α offspring states, $v_{i1}, \dots, v_{i\alpha}$, are non-negative integers. It is now a matter of substitution to verify that $\hat{\mathbf{v}}$ is an approximate eigenvector of \hat{T} . It can be seen that the sum of the state weights of T equals the sum of the state weights of \hat{T} . From (7.13) it follows that no offspring state is heavier than its forebear. Thus, compared with T , either the maximum weight of states of \hat{T} , or the number of states with maximum weight, has been reduced. The splitting process is performed a finite number of rounds with the role of the new T played by \hat{T} until a vector $\hat{\mathbf{v}}$ is reached having all its components equal to zero or one. States having zero weight are removed by crossing out rows and columns of \hat{T} corresponding to the weightless states. Then we have found a finite-state machine having an $\hat{N} \times \hat{N}$ transition matrix \hat{T} , $\hat{N} \leq \sum_i v_i$, that satisfies $\hat{T} \hat{\mathbf{v}} \geq 2^m \hat{\mathbf{v}}$ and, thus, with row sums at least 2^m . The procedure will be illustrated with an example.

Example 7.5 Consider the '0/1' 5×5 transition matrix T , and output function ζ , which are taken from Example 7.3:

$$T = \begin{bmatrix} 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 0 & 0 \\ 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 0 & 0 \end{bmatrix}, \quad \zeta = \begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 0 \\ 1 & 0 & 1 \\ 0 & 1 & 1 \\ 0 & 1 & 0 \end{bmatrix}.$$

The maximum real eigenvalue of the matrix T is $2^{2.08}$, so let $m = 2$. An approximate eigenvector for the eigenvalue $2^m = 4$ is $\mathbf{v}^T = (2, 1, 2, 2, 1)$. It can be verified that indeed $T\mathbf{v} \geq 4\mathbf{v}$. The five-state source will be transformed into an encoder having $\sum_i v_i = 8$ states. State σ_1 has weight $v_1 = 2$, and it has five successors. There are fifteen proper partitions of the set of successors; three of them obey (7.12). Quite arbitrarily we choose $E_1^1 = \{\sigma_3, \sigma_4\}$ and $E_1^2 = \{\sigma_1, \sigma_2, \sigma_5\}$ (it can be verified that conditions (7.12) are satisfied). We split state σ_1 into two offspring states σ_{11} and σ_{12} . Applying (7.13), we find that the weights of the two offspring states are both unity. In similar vein, we split σ_3 into σ_{31} and σ_{32} with successors $\{\sigma_3, \sigma_4\}$ and $\{\sigma_1, \sigma_2, \sigma_5\}$, respectively, and σ_4 into σ_{41} and σ_{42} with successors $\{\sigma_3, \sigma_4\}$ and $\{\sigma_1, \sigma_2, \sigma_5\}$, respectively. After some bookkeeping, we

obtain the 8×8 matrix

$$\hat{T} = \begin{array}{c} 1^{1^1} \\ 2^{1^2} \\ 3^2 \\ 4^{3^1} \\ 5^{3^2} \\ 6^{4^1} \\ 7^{4^2} \\ 8^5 \end{array} \begin{bmatrix} 1^1 & 1^2 & 2 & 3^1 & 3^2 & 4^1 & 4^2 & 5 \\ 0 & 0 & 0 & 1 & 1 & 1 & 1 & 0 \\ 1 & 1 & 1 & 0 & 0 & 0 & 0 & 1 \\ 1 & 1 & 1 & 1 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 1 & 1 & 1 & 0 \\ 1 & 1 & 1 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 1 & 1 & 1 & 1 & 0 \\ 1 & 1 & 1 & 0 & 0 & 0 & 0 & 1 \\ 1 & 1 & 1 & 1 & 1 & 0 & 0 & 0 \end{bmatrix}, \quad \hat{\zeta} = \begin{array}{c} 1^1 \\ 1^2 \\ 2 \\ 3^1 \\ 3^2 \\ 4^1 \\ 4^2 \\ 5 \end{array} \begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 0 \\ 1 & 0 & 1 \\ 1 & 0 & 1 \\ 0 & 1 & 1 \\ 0 & 1 & 1 \\ 0 & 1 & 0 \end{bmatrix}. \quad (7.14)$$

The left hand column gives the new state numbers. The new approximate eigenvector is $\hat{\mathbf{v}}^T = (1, 1, 1, 1, 1, 1, 1, 1)$. In this particular example, we reach, after only one round of splittings, a transition matrix with the virtue that each state has at least $2^m = 4$ branches. The number of successors of states σ_3 and σ_8 exceeds the four required branches per state. They both have five branches. It is therefore legitimate to delete one of the successors of states σ_3 and σ_8 . In principle, the deletions can be made arbitrarily, but sometimes it is more convenient to delete branches such that rows are similar, because states can be merged afterwards. After these operations, we obtain a final \bar{T} whose row sums equal 2^m . Thus we have

$$\bar{T} = \begin{bmatrix} 0 & 0 & 0 & 1 & 1 & 1 & 1 & 0 \\ 1 & 1 & 1 & 0 & 0 & 0 & 0 & 1 \\ 1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 1 & 1 & 1 & 0 \\ 1 & 1 & 1 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 1 & 1 & 1 & 1 & 0 \\ 1 & 1 & 1 & 0 & 0 & 0 & 0 & 1 \\ 1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 \end{bmatrix}$$

It is interesting to reflect upon the fact that before deleting the branches, the transition matrix \hat{T} given by (7.14) still has the maximum real eigenvalue, $2^{2.08}$, of the original transition matrix. After deleting the excess branches, we obtain \bar{T} whose (maximal) eigenvalue is exactly $2^m = 4$.

The ACH algorithm can be summarized as follows:

1. Write down the connection matrix D and determine the capacity $C(d, k)$ of the constrained channel. Choose integers m and n such that $C \geq m/n$.
2. Compute D^n .
3. Verify if $[D]_{ij}^n \in \{0, 1\}$. If so, then $T = D^n$ else find a q th order edge graph $D^{[q]}$ for which $D^{[q]n}$ has zero/one elements only. Then set $T = D^{[q]n}$.
4. Compute an approximate eigenvector \mathbf{v} for T .

5. Use T and \mathbf{v} as inputs for the state splitting procedure. Repeat the state splitting procedure until an eigenvector $\hat{\mathbf{v}}$ is reached having all components equal to zero or one.
6. Delete the excess successor states, and states for which $v_i = 0$. Merge states for encoder simplicity.
7. Assign source words to the branches.

Table 7.15: Transition table $g(\sigma_i, \beta_u)$ of rate 2/3, (0,1) code.

β_u	00	01	10	11
σ_i				
1	4	5	6	7
2	1	2	3	8
3	1	2	3	4
4	4	5	6	7
5	1	2	3	8
6	4	5	6	7
7	1	2	3	8
8	1	2	3	4

After the previous spadework, it is now a simple matter to assemble a rate 2/3, (0,1) encoder. In fact, we already started: Steps 1, ..., 4 of the ACH algorithm were taken in Example 7.3, followed by Steps 4, 5, and 6 in Example 7.5. The last step, Step 7, "Assign source words to the branches" remains to be done.

Table 7.16: Output table $\chi_i = h(\sigma_i)$ of rate 2/3, (0,1) code.

σ_i	$\chi_i = h(\sigma_i)$
1	111
2	111
3	110
4	101
5	101
6	011
7	011
8	010

A Moore-type encoder is defined by its characterizing functions $g(.,.)$ and $h(.,.)$ (see Chapter 2). The next-state function $g(\sigma_i, \beta_u)$, where β_u , $u = 1, \dots, 4$, are the four source words, and the output function $h(\sigma_i)$ are shown in Tables 7.15 and 7.16. The assignment of the source words to the branches has been made rather arbitrarily. The encoder comprises a 3-stage register to store the actual state, and a $(2 + 3) \rightarrow (3 + 3)$ logic array for looking-up the 3-bit codeword and the 3-bit next-state. Other embodiments are, of course, possible. The relationship between the window length of the sliding-block decoder and the various code parameters is still an open problem. For example, for the rate $2/3$, $(0,1)$ code discussed above, we find $r = q = 1$. A more judicious choice of the partitioning of the successor states as described in Example 7.5 would provide a slight improvement in the decoder window length.

7.6 Baldwin codes

Baldwin [22] published an interesting family of rate $1/2$, $(d = 2, \infty)$ codes. The Baldwin codes could be found by invoking the ACH method, but it is much easier to explain it differently. The crux of the Baldwin's code is simple.

7.6.1 Encoder description

All words used in the construction are $d = 2$ constrained. The encoder has two states, called State 1 and 2. The set of codewords in State 1, S_1 , consists of two subsets, denoted by S_{1a} and S_{1b} . Similarly the set of codewords in State 2, S_2 , consists of two subsets, denoted by S_{2a} and S_{2b} . The codewords in the four subsets are characterized as follows.

Table 7.17: Principle of operation of Baldwin's code. An 'x' denotes a don't care."

<i>State 1</i>	<i>next state</i>	<i>State 2</i>	<i>next state</i>
00...xx	1	10...xx	1
00...00	2	01...xx	1
		10...00	2
		01...00	2

Codewords in subset S_{1a} start with two 'zero's. After transmission of a codeword in S_{1a} the encoder remains in State 1. Codewords in S_{1b} start and end with two 'zero's. Note that subsets S_{1a} and S_{1b} have the words

that end with two 'zero's in common. After transmission of a codeword in S_{1b} the encoder will go to State 2. Codewords in subset S_{2a} have a 'one' at one of the two leading bit positions. After transmission of a codeword in S_{2a} the encoder will go to State 1. Codewords in subset S_{2b} have a 'one' at one of the two leading bit positions and they end with 2 'zero's. After transmission of a codeword in S_{2a} the encoder will remain in State 2. An overview of the various subsets and their properties is given in Table 7.17. The number of available codewords in State 1 is $N_{d=2}(n-2) + N_{d=2}(n-4)$, which using (4.2) equals $N_{d=2}(n-1)$. The total number of words available in State 2 is $N_{d=2}(n-2) + N_{d=2}(n-3)$. As $N_{d=2}(n-4) < N_{d=2}(n-3)$, we conclude that the number of words pertaining to State 1 is the smallest, and therefore the size of the code is $N_{d=2}(n-1)$. It is easily seen that the sets of words pertaining to State 1 and State 2 are disjoint (they start with $d=2$ 'zero's or they do not).

The ambiguity of the words that subsets S_{1a} and S_{1b} (or S_{2a} and S_{2b}) have in common can be resolved by observing the two leading bits of the next codeword. This will lead to a very slight error propagation. As an example we worked the Baldwin code for $n=8$. The number of codewords leaving State 1 and State 2 equal 9 and 10 (see Table 4.2, page 55), respectively. After deleting the surplus codewords, we obtained the two-state $(2, \infty)$ code listed in Table 7.18.

Table 7.18: Codebook of two-state $(2, \infty)$ code.

i	$h, g(1, \beta_i)$	$h, g(2, \beta_i)$
0	000000, 1	010000, 1
1	000001, 1	010001, 1
2	000010, 1	010010, 1
3	000100, 1	100100, 1
4	001000, 1	100001, 1
5	001001, 1	100010, 1
6	000000, 2	100000, 1
7	000100, 2	100100, 2

As suggested by Baldwin, his code can be embellished in various ways. The employment of $dklr$ sequences, where $r = \lfloor (k-2)/2 \rfloor$ and $l = k-2-r$, *in lieu* of d sequences leads to a straight forward construction of a (d, k) , $k < \infty$ code. If there are some words left (i.e. more than the required power of two) we can use them, as claimed in Baldwin's patent, as alternative channel representations for reducing the low-frequency components. For example, for $n=16$, Baldwin showed that the two coding tables of State 1 and 2, respectively, are of size 406 and 466, respectively. This leaves

ample spare codewords for the implementation of a rate $8/16$, $(2,11)$ dc-free code. EFMPPlus, a rate $8/16$, $(2,10)$ code, to be discussed in Section 11.5.2, uses four instead of two coding tables, and the same mechanism for reducing the lf components.

7.7 Immink codes

In the next sections we will discuss the construction of simple $d = 1$ and $d = 2$ RLL encoders.

7.7.1 Encoder description, $d=1$ case

In this section, we will describe a finite-state encoder that generates sequences satisfying the $d = 1$ constraint (the k constraint is ignored for a while for ease of presentation). We start with a few ubiquitous definitions. A codeword is a binary string of length n that satisfies the $d = 1$ constraint. The set of codewords, E , is divided into four subsets E_{00} , E_{01} , E_{10} , and E_{11} . The four subsets are characterized as follows.

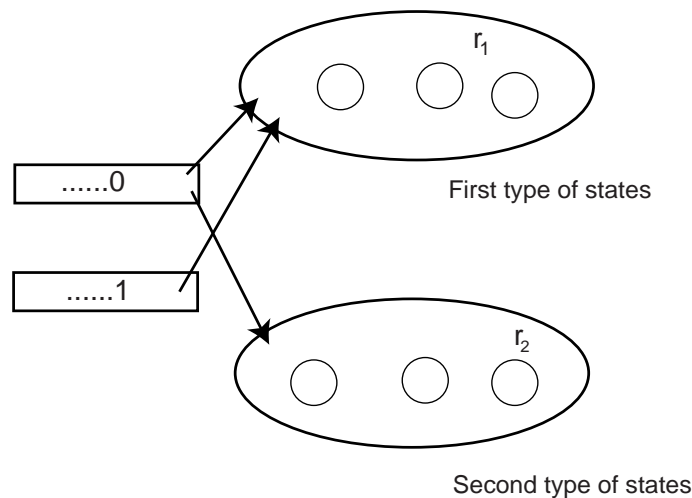


Figure 7.6: Codewords that end with a '0' may be followed by codewords in the r_1 states of the first type and the r_2 states of the second type, while words that end with a '1' may only be followed by codewords in the r_1 states of the first type.

Codewords in E_{00} start and end with a '0', codewords in E_{01} start with a '0' and end with a '1', etc. The encoder has r states, which are divided into two state subsets of a first and second type. The encoder has r_1 states

of the first type and $r_2 (= r - r_1)$ states of the second type. The two types of coding states are characterized by the fact that codewords in the states of the first type must start with a '0', while codewords in states of the second type are free to start with a '1' or a '0'. The encoder state-transition rules are now easily described. Codewords that end with a '0', i.e., codewords in subsets E_{00} and E_{10} may enter any of the r encoder states. Codewords that end with a '1' may be followed by codewords in the r_1 states of the first type only (as, by definition, the codewords in states of the first type start with a '0'). The encoder concept is schematically represented in Figure 7.6. It is essential that the sets of codewords in a given state (of any type) do not have codewords in common (i.e. sets of codewords associated with coding states are disjoint). This property implies that any codeword can unambiguously be identified to the state from which it emerged. Then, as we will show, it is possible to assign the same codeword to different information words (the miraculous multiplication of codewords). Codewords in subsets E_{10} and E_{00} can be assigned r times to different information words, while codewords in subsets E_{11} and E_{01} can be assigned r_1 times to different information words. The decoder can, by observing both the current and the next codeword for identifying the next state, uniquely decide which of the information words was transmitted. Given the above encoder model, we can write down two existential conditions of a rate m/n , $d = 1$ code. Let $|E_{xy}|$ denote the size of E_{xy} . Then, following the above arguments, there are at maximum $r|E_{00}| + r_1|E_{01}|$ codewords leaving the r_1 states of the first type. For a rate m/n code, there should be at least $r_1 2^m$ codewords leaving the r_1 states of the first type. Thus we can write down the first condition

$$r|E_{00}| + r_1|E_{01}| \geq r_1 2^m. \quad (7.15)$$

Similarly, the second condition follows from the fact that there should be a sufficient amount of codewords leaving the r states. We find

$$r(|E_{00}| + |E_{10}|) + r_1(|E_{01}| + |E_{11}|) \geq r 2^m. \quad (7.16)$$

Note that the above inequalities are reminiscent of the approximate eigenvector equation, see (7.1), page 170, which plays an essential role in a variety of code constructions. There is quite a difference as the two inequalities above imply a specific encoder structure at hand, while the approximate eigenvector equation is merely a guide for building a code.

With a small computer we can, given m and n , easily find integers r and r_1 that satisfy the above two conditions. An example of a very small rate $2/3$, $(1, \infty)$ code will exemplify the above.

Example 7.6 Let $m = 2$ and $n = 3$. Then $E_{00} = \{000, 010\}$, $E_{01} = \{001\}$, $E_{10} = \{100\}$, and $E_{11} = \{101\}$. The choice of $r_1 = r_2 = 1$ satisfies Conditions (7.15) and (7.16), and we can simply write down the encoding table 7.19.

Table 7.19: Codebook of two-state rate $2/3$, $(1, \infty)$ code.

i	$h, g(1, \beta_i)$	$h, g(2, \beta_i)$
0	000, 1	100, 1
1	000, 2	100, 2
2	010, 1	001, 1
3	010, 2	101, 1

With the above simple case in mind, we can easily generalize the above two-state encoder to larger values of n . Assume that all codewords in State 1 start with a 'zero', while codewords in State 2 may start with a 'one' or a 'zero'. Then codewords ending with a 'one' are directed to State 2, and codewords ending with a 'zero' are directed to State 1 and State 2. As a result, a two-state ($d = 1$) encoder has a maximum of $2N_1(n - 1) + N_1(n - 2) = N_1(n + 1)$, $n > d + 1$, codewords, where $N_1(n)$ denotes the number of ($d = 1$) sequences of length n . The number of source words that can be accommodated is $\lfloor N_1(n + 1)/2 \rfloor$. For $n = 6$, we have a total of 34 suitable codewords, and by deleting the two all-zero codewords, we may construct a 2-state rate $4/6$, $(1, 10)$ code.

In the next example, we will show the details of the construction of a rate $9/13$, $d = 2$ code.

Example 7.7 Assume the construction of a rate $9/13$ encoder. Then $|E_{00}| = 233$, $|E_{10}| = |E_{01}| = 144$, and $|E_{11}| = 89$. Table 7.20 shows values of r_1 , r_2 , and $r = r_1 + r_2$ that satisfy Conditions (7.15) and (7.16).

Table 7.20: Values of r_1 and r_2 that satisfy Conditions (7.15) and (7.16).

r_1	r_2	$r = r_1 + r_2$
3	2	5
5	3	8
6	4	10
7	5	12
8	5	13

After finding suitable values of r_1 and r_2 , the next step in the code construction is the distribution of the various codewords among the various states. In order to find such a distribution, a trial and error approach has been used. Table 7.21 shows, for $r_1 = 3$ and $r_2 = 2$ as an example (Note that the distribution given is not unique, there are many other ways for allocating the codewords to the states), how the codewords in the various subsets can be allocated to the various states.

From Table 7.21, we discern that the subset E_{00} of size 233 has 72 words in States 1 and 2, 87 words in State 3, and 2 words in State 5. Thus in total: $72+72+87+2=233$. Similarly, it can be verified that the four row sums equal the number of codewords in each of the four subsets. Codewords that end with a '0', i.e., codewords in E_{10} and E_{00} , can be assigned $r = 5$ times to different information words, while codewords that end with a '1', i.e. codeword in E_{11} and E_{01} , can be assigned $r_1 = 3$ times to different information words. Thus, the total number of information words that can be assigned to the codewords in State 1 is $5 \times 72 + 3 \times 52 = 516$. Similarly, it can be verified that from any of the $r = 5$ encoder states there at least 516 information words that can be assigned to codewords, which shows that the code can accommodate 9-bit information words. An enumeration table such as Table 7.21 suffices to construct a code by assigning codewords to the coding states and source words.

Table 7.21: Distribution of the various subsets and states.

group/state	1	2	3	4	5
E_{00}	72	72	87	0	2
E_{01}	52	52	27	0	13
E_{10}	0	0	0	72	72
E_{11}	0	0	0	52	37

It can be verified with the procedure outlined above that a 13-state encoder of (code) size 520 can be created. The maximum size of any 13-bit $(1, \infty)$ code equals $\lfloor 2^{13C(1, \infty)} \rfloor = 521$, and we therefore conclude that the above code is quite efficient ($\eta = 0.9996$) particularly considering that the encoder has a relatively small number, 13, of states. This code is supposedly one of the most efficient in existence in terms of relative performance. Such extremely efficient codes could up till now only be constructed with "large" codewords, but as shown here also selected "small" codes can have a rate which is very close to the channel capacity.

As the above code can accommodate more than the required 512 words, surplus 'worst-case' codewords can be deleted for minimizing the k constraint. After a judicious process of deleting codewords that end or start with "long" runs of '0's, we constructed a 5-state (1,18) code, and a 13-state (1,14) code. Note in Table 4.4, page 60, that the smallest possible k for a rate 9/13 code equals 12.

The code can be decoded by a sliding block decoder, see Section 7.2, of window size two. Single channel bit errors can thus lead to at most two decoded m -bit symbols. The decoder comprises two look-up tables: the next-state look-up table and the data look-up table. The next-state look-up table has the next codeword as an input, and the state to which this word belongs as an output. The data look-up table has the output of the next-state look-up table and the current codeword as an input, and the output of the data look-up table is the decoded information word.

7.7.2 Encoder description, $d=2$ case

In this section we will describe a finite-state encoder that generates sequences that satisfy the $d = 2$ constraint (note that the k constraint will be ignored for a while). We start with a few definitions. The encoder is assumed to have r states, which are divided into three state subsets of states of a first, second, and third type. The state subsets are of size r_1 , r_2 , and $r_3 (= r - r_1 - r_2)$, respectively. A codeword is a binary string of length n that satisfies the $d = 2$ constraint. The set of codewords is divided into nine subsets denoted by E_{0000} , E_{0001} , E_{0010} , E_{0011} , E_{0100} etc, where the two first symbols of the subset subscript denote the first two symbols of the codeword, and the last two symbols of the subset subscript denote the last two symbols of the codeword. Thus, codewords in E_{0000} start and end with '00'; codewords in E_{0001} start with '00' and end with a '01', etc. The codewords in the various subsets are distributed over the various states of the three types such that

- codewords in states of the first type start with '00',
- codewords in states of the second type start with '01' or '00', and
- codewords in states of the third type start with '10', '01' or '00'.

The state-transition rules are now easily described. Codewords that end with the string '00', i.e., codewords in subsets E_{0000} , E_{0100} , and E_{1000} may enter any of the r encoder states. Codewords that end with a '10' may not be followed by codewords in a state of the third type. Similarly, codewords that end with a '1' may only be followed by codewords belonging to states of the first type. The state sets of codewords from which a selection is to be made do not have codewords in common. As a result, it is possible to assign the same codeword to different information words. For example, codewords that end with '00', i.e. codewords in subsets E_{0000} , E_{0100} , and E_{1000} , may enter any state so that these codewords can be assigned $r = r_1 + r_2 + r_3$ times to different information words. Codewords that end with '10', i.e. words in subsets E_{0010} , E_{0110} , and E_{1010} may enter states of the 1st and 2nd type so that these codewords can be assigned $(r_1 + r_2)$ times to different information words. Similarly, codewords that end with a '1', i.e. words in the remaining subsets E_{0001} , E_{0101} , and E_{1001} can be assigned r_1 times. Note that the Baldwin code, see Section 7.6, can be seen as a special case where $r_2 = 0$. Given the above encoder model, it is straightforward to write down three conditions for the existence of such a rate m/n code.

$$r|E_{0000}| + (r_1 + r_2)|E_{0010}| + r_1|E_{0001}| \geq r_1 2^m, \quad (7.17)$$

$$r|E_{0100}| + (r_1 + r_2)|E_{0110}| + r_1|E_{0101}| \geq r_2 2^m, \quad (7.18)$$

$$r|E_{1000}| + (r_1 + r_2)|E_{1010}| + r_1|E_{1001}| \geq r_3 2^m. \quad (7.19)$$

In a similar vein as with the ($d = 1$) codes discussed previously, we have experimented with the selection of suitable values of m , n , r_1 , r_2 , and r_3 . Many good codes have been found. As a typical example, which is amenable for a hand check, we will show results of a 9-state, $(2, k)$ code of rate 6/11.

Table 7.22: Example of the distribution of the various subsets and states of a rate 6/11, $(2, k)$ code.

group/state	1	2	3	4	5	6	7	8	9
E_{0000}	6	4	4	4					
E_{0010}	1	2	2	4					
E_{0001}	1	4	4	1			1		
E_{1000}					4	4	5		
E_{1010}					2	2	2		
E_{1001}					4	4	1		
E_{0100}								4	5
E_{0110}								2	2
E_{0101}								4	2

Given the choice of the code rate, we use a small computer program to find suitable values of r_1 , r_2 , and r_3 that satisfy conditions (7.17), (7.18), and (7.19). A possible distribution of the various codeword sets, where we opted for $r_1 = 4$, $r_2 = 2$, and $r_3 = 3$, is shown in Table 7.22. Such a distribution table suffices to construct the code.

Table 7.23: Code size, M , for $d = 2$, $n = 16$, and selected values of the number of encoder states r .

r_1	r_2	r_3	r	M	$\eta = R/C(2, \infty)$
1	1	1	3	426	0.98995
2	1	2	5	430	0.99148
3	1	3	7	431	0.99186
4	2	3	9	447	0.99782
7	3	5	15	450	0.99891
13	6	9	28	452	0.99963

After judiciously barring worst-case codewords from the coding table, it is possible to construct a rate 6/11, $(2, 15)$ code. Note, see Table 4.4, page 4.4, that $k = 11$ is the smallest value possible for the given rate 6/11.

Using the above method, it is possible to construct a 9-state rate $11/20$, $(2,23)$ code, whose efficiency is 0.25% less than capacity. In addition, a rate $7/13$, $(2,11)$ code was constructed, whose efficiency is 1.1% less than capacity. The efficiency of the new construction technique can be exemplified by an example, where the code size, M , does not equal a power of two. The spare codewords can obviously be used as an alternative channel representation for suppressing the lf components. Table 7.23 shows the efficiency $\eta = R/C(2, \infty)$ as a function of the number of encoder states r . It shows that the construction technique is extremely efficient reaching efficiencies that are only a tenth of a percent below capacity. Note that the maximum size of a code with codewords of length $n = 16$ equals 453.

7.7.3 Very efficient coding schemes

Many good RLL codes have been published, but information recording has a constant need for enhancing the information density on the record carrier, and a possible solution to this end is an increase of the rate of the code. As shown in Chapter 4, the tenets of information theory set a limit to the maximum rate of RLL constrained codes. Table 4.4, page 60, tabulates $C(d, k)$ as a function of d and k . We may observe, for example, that for $d = 1$ and $k = 7$ the Shannon capacity, $C(1, 7)$, has a value of 0.6793. As a consequence, an encoder that translates arbitrary sequences into sequences that have at least $d = 1$ and at most $k = 7$ 0's between successive 1's, cannot have a rate larger than 0.6793. Rate $2/3$, $(1,7)$ codes have been used in recording systems for more than a quarter of a century, see for example this chapter. Recently, embellishments of this classic code have been published, which turn the RLL code into a DCRL code [181, 257]. Alternatively, Hassner *et al.* presented rate $2/3$, $(1,9)$ and $(1,13)$ RLL codes in which predetermined RLL-coded sequences are inhibited from indefinite recurrence by imposing an additional maximum transition run (MTR) constraint [123]. For ease of presentation we will first focus on the design of RLL codes with $d = 1$. Later we will extend the ideas to the design of codes with $d = 2$. The code rate, $2/3$, of the $(1,7)$ code is slightly less than the Shannon capacity, 0.6793, and the code is therefore an efficient one. The rate efficiency $\eta = R/C(d, k)$ of the rate $2/3$, $(1,7)$ code is $0.6667/0.6793 = 0.981$, which reveals that we can at most gain 1.9% in rate by an alternative, more efficient, code redesign. There are only two approaches for constructing a $(1, k)$ RLL code, whose rate is larger than two-thirds. Firstly, we may relax the maximum runlength k to a value larger than 7. Note that a $(1,7)$ code was first put to practical use in the early seventies, and that since the advent of hard-disk drives (HDDs), significant improvements in signal processing for timing recovery circuits have made it possible to employ codes with a much larger maximum runlength k . Secondly, on top of that, we

may endeavor to design a more efficient code. Note that by fully relaxing the k constraint, i.e. set $k = \infty$, we can at most gain 3.97% in code rate. In other words, a viable improvement in code rate of a ($d = 1$) encoder ranges from 1.9 to 3.97%.

In the sequel of this section, we will show how to create a (1,14) code, whose rate is 3.85% better than the traditional rate $2/3$, (1,7) code. We start, in the next subsection, with a simple problem, namely finding integers m and n that improve the rate, $2/3$, of the industry standard code.

Suitable integers m and n for $d = 1$

We will start with a simple, but very illuminating exercise, namely a search for pairs of integers m and n that are suitable candidates for a coding rate exceeding $2/3$. Obviously, the "best" code is a code with a rate, m/n , that exactly equals the capacity $C(d, k)$ for desired values of d and k . One is tempted to ask if it is possible to choose m and n such that $m/n = C(d, k)$. The answer, a sounding no, was given by Ashley & Siegel, see Theorem 4.1, page 62, who showed that, besides a very few trivial exceptions, the capacity, $C(d, k)$, is an irrational number. As the rate of a code m/n , m and n integers, is rational, the capacity can only be approached.

Table 7.24: Integers m and n such that $2/3 < R = m/n < C(1, \infty)$. The quantity $\eta = R/C(1, \infty)$ expresses the code efficiency.

m	n	$1 - \eta$ %
34	49	0.0525
9	13	0.2786
11	16	0.9711
13	19	1.4449
15	22	1.7895
17	25	2.0514

In order to obtain some feeling if there are many "practical" pairs of such integers m and n , we wrote a one-line computer program for searching integers m and n that satisfy the inequalities $2/3 < m/n < C(1, \infty)$, where for reasons of implementation we set $n < 50$. All pairs of integers found are shown in Table 7.24. Surprisingly there are just six m and n pairs whose quotient is larger than $2/3$. Perusal of the table reveals that the code rate $m/n = 9/13$ is highly attractive as it is just 0.28% below the Shannon capacity $C(1, \infty)$. The next better code of rate $34/49$ is far less attractive as it is much more complex and adds a minute 0.2% to the density gain with respect to a rate $9/13$ code.

Suitable integers m and n for $d = 2$

RLL codes with minimum runlength parameter $d = 2$ have been widely published. Table 4.4 tabulates $C(2, k)$ as a function of k , and from this table the reader can easily discern the head room available for the design of a code of rate $R = m/n > 8/15$. The rate $8/15$ is, see Table 4.4, 3.3% below channel capacity $C(2, \infty)$. Table 7.25 shows selected values of m and n , where $8/15 \leq m/n < C(2, \infty)$ and $n < 50$. The pairs of integers are ranked by their corresponding efficiency $R/C(2, \infty)$.

Table 7.25: Integers m and n such that $8/15 < R = m/n < C(2, \infty)$. The quantity $\eta = R/C(2, \infty)$ expresses the code efficiency.

m	n	$1 - \eta$ %
11	20	0.2720
17	31	0.5644
6	11	1.0962
19	35	1.5672
13	24	1.7830
20	37	1.9872
7	13	2.3642
15	28	2.8623
8	15	3.2940

Clearly, the quotients $11/20$, $6/11$, and $7/13$ are suitable candidate rates for the creation of small ($d = 2$) codes. Efficiency-wise speaking the code of rate $17/31$ looks more attractive, but the code is far too complex for an implementation using state of the art technology. Table 7.26 summarizes $d = 1$ and $d = 2$ (d, k) codes, which have been found with the methods presented above [164]. Note that the efficiency of the majority of the new codes is just a few tenths of a percent below capacity. Kim [204] has been granted a U.S. Patent on an alternative embodiment of a rate $7/13$, (2,25) code, which is based on the 3PM method, see Section 5.8.1, page 127.

7.8 Discussion

Many other examples of sliding-block codes have been published in the (patent) literature. Four low-rate recording codes have been presented by Adler *et al.* [4], and Van Rensburg & Ferreira [289]. A rate $2/5$, (3,11) code was published by Lee [219]. In [3], a rate $2/3$, (1,7) code is disclosed by Adler *et al.*, which is similar to the look-ahead code designed by Jacoby & Kost [174]. Examples of rate $1/2$, (2,10) codes have been presented by

French & Wolf [103]. A code designed by Adler [5] with properties rate $2/3$, $(1,6)$ code, has the virtue that one error in the encoded string can result in no more than eleven errors in the decoded data.

Table 7.26: Survey of newly developed codes. (Taken from [164].)

m	n	d	k	states	$\eta = R/C(d, k)$
11	20	2	23	9	0.9975
7	13	2	11	9	0.9880
6	11	2	15	9	0.9915
9	13	1	14	13	0.9979
9	13	1	18	5	0.9973
11	16	1	10	13	0.9951

Hollmann [139], Chapter 4, presented a code with the same parameters, but the window is shorter, namely five 3-bit codewords, and error propagation of nine data bits only. A blind application of the ACH algorithm will not provide a decoder with the shortest decoder window length. Indeed, every step of the ACH algorithm may affect the complexity of the encoder and decoder. The initial choice of the approximate eigenvector, the state splitting procedure, which excess successor states are to be deleted, and the eventual assignment of source words to branches are crucial as all these steps of the algorithm govern to some extent the final complexity and the decoder window length. It should be underlined that there are no systematic algorithms available to find an encoder/decoder pair of minimal hardware, or to find an encoder algorithm with a corresponding decoder of the shortest decoder window length. Complexity issues have been investigated by many workers. For example, the minimum number of encoder states has been dealt with by Marcus & Roth [234], and the minimum sliding-block window has been studied by Ashley, Marcus, and Roth [9, 14, 16] and Hollmann [139]. Ruckenstein & Roth studied the anticipation of encoders for input-constrained channels [293]. Algorithms for searching the sliding-block decoder mapping of minimum window length were published by Kamabe [183]. Variable-length state splitting was introduced by Heegard, Marcus & Siegel [129].

As can be seen from the many examples given in the last part of this chapter, numerous construction techniques have been developed over the last forty years. A priori, it is hard to say which of these techniques is 'best'.

Variable-length codes, look-ahead codes, or codes designed by the state-splitting (ACH) algorithm can, as we have demonstrated, yield dramatic reduction in complexity of the encoder and decoder for codes of certain rates

as compared with their block code counterparts. The approaches mentioned lead to codes of approximately the same complexity. The exact relationship between the methods is still an open problem [87]. The relationship between codes designed by the ACH versus bounded-delay codes was investigated by Hollmann [137, 139, 138]. The design procedures can be made completely systematic, in the sense that computer programs can (and probably have been) written to automatically generate coding and decoding tables. The eventual design may benefit from some interaction with a human operator who can steer the algorithm.

As we have demonstrated, notably codes with rate $1/2$ or $2/3$ can be designed very efficiently with the ACH (and other) algorithm, and it is scarcely conceivable that one could improve their performance and/or hardware requirements. If, however, the code rate is of the form m/n , m and n large, fixed-length block codes using, for example, $(dklr)$ sequences may offer a more advantageous solution since it is easier to preserve a particular mapping between the source and the code symbols. Look-up tables or alternatively enumerative coding, see Chapter 6, are attractive methods for translating the words. The quest for higher code efficiencies has given impetus to alternative constructions, such as block codes having very long codewords, which approach channel capacity within 0.5% for a block length of 400 to 500 bits (see Chapter 6). In some instances a comprehensive word assignment can be discovered that allows the use of Boolean equations for encoding and decoding, as in the case of EFM (Section 5.6.1, page 114), the rate $8/9$, $(0,3)$ block code (Section 5.6.2, page 114), or other k -constrained codes (Section 5.6.3, page 115). An additional advantage of fixed-length block codes is the convenience with which they can be incorporated into a fixed-frame format having synchronization patterns at regular distances.

Chapter 8

Dc-balanced Codes

8.1 Introduction

Binary sequences with spectral nulls at zero frequency have found widespread application in optical and magnetic recording systems. *Dc-balanced*, *dc-free*, or *spectral null* codes, as they are often called, have a long history and their application is certainly not confined to recording practice. In digital transmission, it is sometimes desirable for the channel stream to have low power near the zero frequency. Since the early days of digital communication over cable, dc-balanced codes have been employed to counter the effects of low-frequency cut-off due to coupling components, isolating transformers, and so on. In optical recording, dc-balanced codes are employed to circumvent or reduce interaction between the data written on the disc and the servo systems that follow the track. Low-frequency disturbances, for example due to fingerprints, may result in erroneous retrieval of the recorded data. Errors of this type are avoided by high-pass filtering, which is only permissible provided the encoded sequence itself contains no low-frequency components, or, in other words, has a spectral null at the zero frequency (dc). Many of the most important block codes for spectral shaping purposes fall into the category of dc-balanced codes. Codes with spectral nulls at other frequencies than zero are often used in recorder systems for tracking and pilot tone insertion [148, 233].

Practical coding schemes devised to achieve suppression of low-frequency components are mostly constituted by block codes. More advanced coding techniques, such as look-ahead encoding, or bounded-delay encoding, as discussed in previous chapters, have not been used for generating dc-balanced sequences. The reason is, probably, that dc-balanced codes usually have a small redundancy. A code rate of, say, 24/25 or higher, is a desirable objective, and the design challenge of such high-rate codes is focused on the complexity of the encoding and decoding tables.

As standard practice, the source digits are grouped in source words of m

digits, which are translated, using a conversion table, known as codebook, into blocks of n digits. The approaches which have actually been used in practice for dc-balanced code design are basically four in number:

Zero-disparity code,
 Low-disparity code,
 Polarity bit code,
 Guided scrambling.

The *disparity* of a codeword is defined as the excess of the number of 'one's over the number of 'zero's in the codeword. Thus the codewords '000110' and '100111' have disparity -2 and +2, respectively. A zero-disparity codeword contains equal numbers of 'one's and 'zero's. The obvious method for constructing dc-balanced codes is to employ zero-disparity codewords that have a one-to-one correspondence with the source words, and therefore the code is state independent.

A logical step, then, is to extend this mechanism to family of the *low-disparity* bi-mode codes, where the translations are not one-to-one. The zero-disparity codewords are uniquely allocated to the source words as in the zero-disparity code. The other, non-zero-disparity, codewords are allocated in pairs of opposite disparity. Each of the two representations, *modes*, 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 accumulated disparity, or the *running digital sum*, of the encoded sequence, after transmission of the new non-zero-disparity codeword, is as close to zero as possible. The running digital sum (RDS) is defined for a binary stream as the accumulated sum of 'one's and 'zero's (a 'zero' counted as -1) counted from the start of the transmission. Both of the basic approaches to dc-balanced coding are due to Cattermole [52, 53] and Griffiths [117, 118].

The third coding method was devised by Bowers [42] and Carter [50, 51] who proposed a slightly different construction of dc-balanced codes as being attractive because no look-up tables are required for encoding and decoding. They proposed a code where $(n - 1)$ (binary) source symbols are supplemented by one symbol called the *polarity bit*. The encoder has the option to transmit the n -bit codewords without modification or the encoder may invert all, n , symbols. Again, like in the low-disparity code, the choice of a specific translation is made in such a way that the accumulated disparity is as close to zero as possible. The polarity bit is used by the decoder to identify whether the transmitted codeword has been inverted or not.

The study of the fourth method, *guided scrambling*, was given new impetus by the work of Fair *et al.* [76, 77]. Guided scrambling is a member of a larger class of related coding schemes, called *multi-mode* code, where each source word can be represented by a member of a (usually very large) se-

lection set of codewords. The encoder opts for transmitting that codeword that minimizes, according to a criterion to be defined, the low-frequency spectral contents of the encoded sequence. Two key ingredients, namely the mapping between source words and codewords plus the selection of the "best" word, require special attention. The spectral (or other) performance of the code greatly depends on both issues. A detailed presentation of guided scrambling will be given in a separate chapter, namely Chapter 10.

The outline of this chapter is as follows. In Section 8.2, we will study the relationship between the capacity, notch width, and digital sum variation of maxentropic dc-balanced sequences. In order to get some insight into the efficiency of the aforementioned construction techniques, we shall evaluate the spectral properties of their respective code streams. The theory provided in Chapter 3 furnishes efficient procedures for the computation of the power spectral density function of block-coded signals produced by an encoder that can be modelled by a finite-state machine. Thus the prospect of numerical analysis need not be depressing. The difficulty is rather that we prefer to know the dependence of the code's performance upon the various design parameters and such dependencies are hard to assess by numerical analysis. Fortunately, the structure of the above simple dc-balanced codes allows us to derive simple expressions for the rate and power spectral density functions. In Section 8.4, the power density function of low-disparity-based channel codes is established. The variance of the running digital sum (in short, sum variance) is, as we will see shortly, a useful concept in evaluating the spectral range of suppressed components. The sum variance of selected channel codes is calculated in Section 8.4.

The theoretical development in Section 8.2 demonstrates that there is a relationship between the performance of the code expressed in the notch width and the redundancy of the code. Coding efficiency and ease of implementation are to a large extent incompatible and we are, for the moment, solely concerned with their mathematical properties. One question of significance is that of code efficiency expressed in terms of spectral notch width and redundancy. The performance of maxentropic dc-balanced sequences, whose properties are studied in Section 8.2, sets a standard by which the performance of dc-balanced code implementations may be measured. In Section 8.6.1, invoking the theory developed in Section 8.2, we intend to appraise the performance of the implemented channel codes. Engineering examples of rate 8/10, 8B10B codes are discussed in Section 8.7.

The design of dc-free codes, where the codeword length is even, is simpler than one where the codewords are of odd length. In Section 8.8, we will outline the challenging task of designing dc-free codes with odd codeword length. A redefinition of the running digital sum will make it possible to efficiently design such codes. Thereafter, in Section 8.9, we will present a simple method for encoding zero-disparity codewords. The codeword can be

obtained by inverting only one symbol in the source word. The information regarding the position, where the symbol has been inverted, is transmitted as a zero-disparity prefix. This method, which is capable of handling (very) large blocks was described by Knuth [206]. A patent was granted in 1982 to Henry describing a similar method [131].

8.2 Preliminaries

Common sense tells us that a certain rate has to be sacrificed in order to convert arbitrary data into a dc-balanced sequence. The first question to be addressed in this chapter is the quantification of the maximum rate, that is capacity, of a sequence given the fact that it contains no low-frequency components. The mathematical tools that will be used to compute the capacity of the dc-balanced channel are derived in a straightforward fashion from the theory developed in Chapter 2. Early work on this topic has been reported by Chien [59] and Justesen & Hoholdt [177, 179]. A description will be provided of statistical characteristics of dc-free sequences generated by a Markov information source having maximum entropy. Knowledge of such ideal, *maxentropic*, sequences with a spectral null at dc is essential for the understanding of the basic trade-offs between the rate of a code and the amount of suppression of low-frequency components that can be achieved at maximum. The results obtained in this section will be exploited to derive a figure of merit of implemented dc-balanced codes that takes into account both the redundancy and the emergent frequency range with suppressed components.

We start with the definition of the running digital sum of a binary sequence. The running digital sum of a sequence, in short, *RDS*, plays a significant role in the analysis and synthesis of codes whose spectrum vanishes at the low-frequency end. Let

$$\{x_i\} = \{\dots, x_{-1}, x_0, \dots, x_i, \dots\}, x_i \in \{-1, 1\}$$

be a binary sequence. The (running) digital sum z_i is defined as

$$z_i = \sum_{j=-\infty}^i x_j = z_{i-1} + x_i. \quad (8.1)$$

Figure 8.1 portrays the various signals defined above. It is assumed in this diagram that the coded input signal in NRZI representation is translated into the write signal using a change-of-state encoder. As spectral null codes are used to tailor the spectrum of the encoded sequence, it will not be too surprising that frequency-domain analysis of encoded sequences play an important role in this section. Frequency-domain analysis of constrained

sequences is based upon the *average power spectral density* or, as it is often called the *power spectrum*.

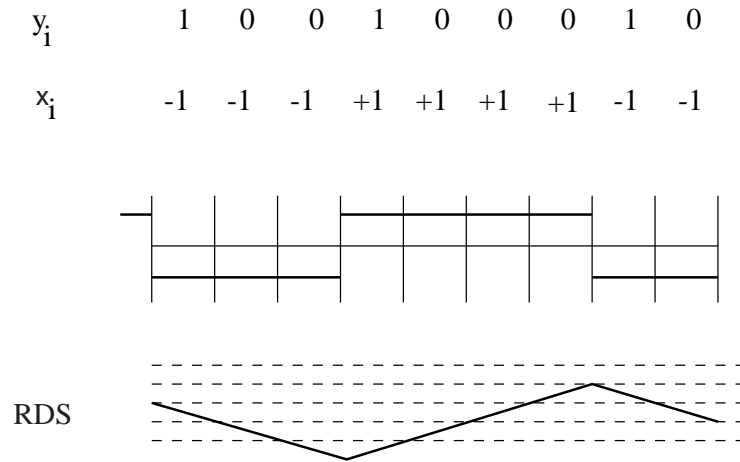


Figure 8.1: Running digital sum (RDS) versus time. Coded input symbols (NRZI) are translated into the write signal (NRZ) using a precoder, and the channel bits x_i . In this example, the RDS assumes at most seven values.

The power spectrum is given by the Fourier transform of the auto-correlation function of the sequence (see Chapter 3). Alternatively, see (3.12), page 31, we can express the power spectrum $H(\omega)$ as

$$H(\omega) = \lim_{M \rightarrow \infty} E \left[\frac{1}{2M+1} \left| \sum_{m=-M}^M x_m e^{-jm\omega} \right|^2 \right], \quad (8.2)$$

where the expectation operator $E\{\cdot\}$ is the expected value over the ensemble of sequences $\{x_i\}$. If the running digital sum, z_i , is bounded, the power spectrum, $H(\omega)$ vanishes at dc. The proof is fairly straightforward. Since, by assumption, z_i is bounded, we have

$$\left| \sum_{m=-M}^M x_m \right| \leq B, \text{ for some } B < \infty. \quad (8.3)$$

Hence,

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

and

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

Pierobon [284] showed that the power density function of an encoded sequence $\{x_i\}$ vanishes at zero frequency if, and only if the encoder is a finite running digital sum encoder.

8.2.1 Capacity of dc-constrained sequences

It will be apparent to the reader, even before matters of implementation are considered, that spectral shaping of a sequence by removing the low-frequency components can be accomplished only at the price of a certain rate loss. In order to provide an answer to such a fundamental question as the rate loss incurred, we shall discuss in detail the work of Chien [59].

Chien studied bipolar sequences $\{x_i\}$, $x_i \in \{-1, 1\}$, that assume a finite number of sum values, that is, at any instant i the RDS z_i of such a sequence meets the condition

$$N_1 \leq z_i \leq N_2,$$

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

$$N = N_2 - N_1 + 1, \tag{8.4}$$

is often called the *digital sum variation* (DSV). A question of vital importance is: How much loss in information rate does one incur by requiring the running digital sum of a sequence to stay within certain limits? Chien addressed the above problem of establishing the information capacity of (z) sequences as a function of the digital sum variation N . In essence, the solution to this problem is provided in Chapter 2. We commence by restating the previous channel constraints in terms of the connection matrix of a finite-state machine.

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

$$\begin{aligned} d_N(i+1, i) = d_N(i, i+1) = 1, \quad i = 1, 2, \dots, N-1, \\ d_N(i, j) = 0, \quad \text{otherwise.} \end{aligned} \tag{8.5}$$

A matrix, such as D_N , having ones in the upper- and lower-diagonals and zeros elsewhere is called a *symmetric Toeplitz matrix*. The process discussed above is also known as the random-walk problem with reflecting

walls. Figure 8.2 portrays the Mealy-type along with the equivalent Moore-type finite-state transition diagram for a sequence that assumes $N = 4$ digital sum values.

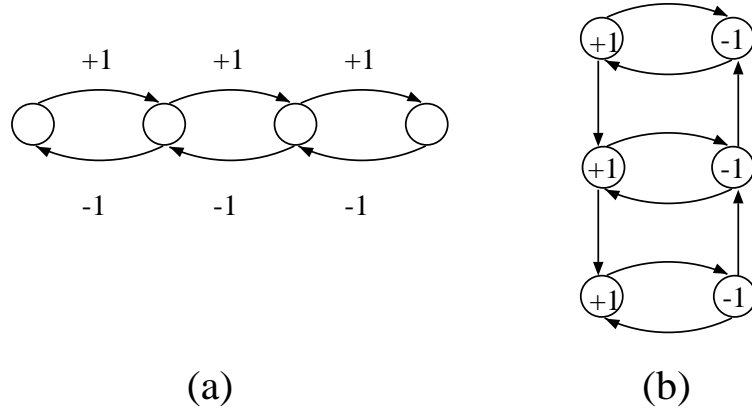


Figure 8.2: (a) State transition diagram of a four-state Mealy-type finite-state machine that generates a sequence with digital sum variation $N = 4$. The states, shown as circles, are connected by arrows which represent allowed transitions from one state to the other. Along the edges we have indicated the symbols emitted by the machine when the chain goes from one state to the other. (b) A six-state Moore-type representation of the same channel.

In the Mealy-type finite-state machine the states, shown as circles, are connected by arrows which represent admissible transitions from one state to the other. Along the edges we have indicated the symbols emitted by the machine when the chain goes from one state to the other. The equivalent Moore-type finite-state machine (see Figure 8.2b) has six states. The information-source model enables us to compute the Shannon capacity of the constrained channel. As was established in Chapter 2, the capacity equals the base-2 logarithm of the largest real eigenvalue of the connection matrix.

It is usually not possible to find a simple, closed-form, expression for the capacity, and one has to rely on numerical methods. This instance is an interesting exception to the rule, as the structure of D_N allows us to provide a closed-form expression for the capacity of an RDS-constrained channel. To this end, let

$$\Phi_N(z) = \det[zI - D_N] \quad (8.6)$$

designate the characteristic polynomial of D_N . The first few characteristic

polynomials $\Phi_N(z)$ can be evaluated by hand:

$$\begin{aligned}\Phi_1(z) &= z, \\ \Phi_2(z) &= z^2 - 1, \\ \Phi_3(z) &= z^3 - 2z, \\ \Phi_4(z) &= z^4 - 3z^2 + 1.\end{aligned}$$

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

$$\Phi_N(z) = z\Phi_{N-1}(z) - \Phi_{N-2}(z), \quad N = 3, 4, \dots \quad (8.7)$$

Equation (8.7) holds for $N = 2$ provided we define $\Phi_0(z) = 1$. To prove (8.7), we expand the determinant in (8.6) with respect to the first column. The eigenvalues of the matrix D_N , which are denoted by $\lambda_1, \dots, \lambda_N$ are the zeros of $\Phi_N(z)$. The associated eigenvectors, denoted by \mathbf{v}_i , $i = 1, \dots, N$, are

$$\mathbf{v}_i = (\Phi_0(\lambda_i), \Phi_1(\lambda_i), \dots, \Phi_{N-1}(\lambda_i))^T. \quad (8.8)$$

Introducing the notation $z = 2 \cos x$, (8.7) becomes

$$\Phi_N(2 \cos x) = (2 \cos x)\Phi_{N-1}(2 \cos x) - \Phi_{N-2}(2 \cos x), \quad N = 2, 3, \dots,$$

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

$$\Phi_N(2 \cos x) = a_1 e^{jNx} + a_2 e^{-jNx},$$

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

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

or

$$\Phi_N(z) = \frac{\sin[(N+1) \cos^{-1}(z/2)]}{\sin[\cos^{-1}(z/2)]}. \quad (8.9)$$

The zeros of $\Phi_N(z)$ are, as can easily be seen in (8.9),

$$\lambda_i = 2 \cos \frac{i\pi}{N+1}, \quad i = 1, \dots, N. \quad (8.10)$$

Thus, the maximum real eigenvalue, i.e. the maximum zero of (8.9), of the matrix defined by (8.5) is given by the simple expression

$$\lambda = \max\{\lambda_i\} = 2 \cos \frac{\pi}{N+1}, \quad (8.11)$$

and thus the capacity of the z -constrained channel is

$$C(N) = \log_2 \lambda = \log_2 2 \cos \frac{\pi}{N+1}, \quad N \geq 3. \quad (8.12)$$

Table 8.1 lists the capacity $C(N)$ and sum variance versus the digital sum variation N , which are deduced using (8.12) and (8.31). The quantity called sum variance $\sigma_z^2(N)$ will be discussed in the next section. It can be seen that the results shown in Table 8.1 are intuitively correct for extreme values of N . If $N \rightarrow \infty$ there is a greater degree of freedom to allow sequences which, of course, entails a vanishingly small amount of rate loss.

Table 8.1: Capacity and sum variance of maxentropic (z) sequences versus digital sum variation N .

N	$C(N)$	$\sigma_z^2(N)$
3	0.5000	0.5000
4	0.6942	0.8028
5	0.7925	1.1667
6	0.8495	1.5940
7	0.8858	2.0858
8	0.9103	2.6424
9	0.9276	3.2639
10	0.9403	3.9506
11	0.9500	4.7026

It can be seen that the sum constraint is not very expensive in terms of rate loss when N is relatively large. For instance, a sequence that takes at maximum $N = 8$ sum values has a capacity $C(8) = 0.91$, which implies a rate loss of less than 10%.

8.2.2 Spectra of maxentropic sequences

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

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

where A is the diagonal matrix $A = \text{diag}(\hat{v}_1, \dots, \hat{v}_N)$. Since D_N is symmetric, the stationary probability π_i pertaining to state σ_i is given by

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

where ρ is chosen to retain the normalizing condition

$$\sum \pi_i = 1.$$

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

$$\pi_i = \frac{2}{N+1} \sin^2 \frac{\pi i}{N+1}, \quad i = 1, 2, \dots, N. \quad (8.13)$$

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

Example 8.1 Consider a sequence with digital sum variation $N = 5$. The characteristic equation of

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

is

$$z(z^2 - 1)(z^2 - 3) = 0,$$

so that the maximum real eigenvalue is $\lambda = \sqrt{3}$. The capacity of the constrained sequence is

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

which coincides with (8.12) and Table 8.1. The corresponding eigenvector is

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

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

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

The stationary probability distribution vector is

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

Associated with the unifilar Markov information source is the auto-correlation function of the running sum $\{z_i\}$:

$$R_z(k) = E\{z_i z_{i+k}\}.$$

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

$$H_z(\omega) = \sum_{k=-\infty}^{\infty} R_z(k) e^{-jk\omega} = R_z(0) + 2 \sum_{k=1}^{\infty} R_z(k) \cos k\omega. \quad (8.14)$$

For the sake of convenience, we shall assume $E\{z_i\} = 0$. The auto-correlation function of the sequence $\{z_i\}$, may be expressed as (see Chapter 3)

$$R_z(k) = \zeta^T \Pi Q^{|k|} \zeta, \quad (8.15)$$

where ζ^T indicates the row vector

$$\zeta^T = (\zeta(\sigma_1), \zeta(\sigma_2), \dots, \zeta(\sigma_N))$$

and Π is the diagonal matrix $\Pi = \text{diag}(\pi_1, \dots, \pi_N)$. Since it is assumed that the values of $\{z_i\}$ are centered around zero, we obtain

$$\zeta(\sigma_i) = i - \frac{N+1}{2}, \quad i = 1, 2, \dots, N.$$

The preceding expressions are not very suitable for numerical purposes. The following analysis is more convenient in that respect.

The $N \times N$ state-transition matrix Q has N distinct eigenvalues $\{1, \mu_2, \dots, \mu_N\}$ and corresponding left eigenvectors $\mathbf{u}_1 = \boldsymbol{\pi}, \mathbf{u}_2, \dots, \mathbf{u}_N$. The eigenvectors are distinct and constitute a basis, so that

$$R_z(k) = \zeta^T \Pi Q^{|k|} \zeta = \left(\eta_1 \boldsymbol{\pi} + \sum_{j=2}^N \mu_j^{|k|} \eta_j \mathbf{u}_j \right) \zeta. \quad (8.16)$$

The constants η_i , $i = 1, \dots, N$, can be found by evaluating $R_z(k)$, $k = 0, \dots, N-1$.

After having established an expression for the auto-correlation function of the running sum sequence $\{z_i\}$, it is a simple matter to find a relation for the auto-correlation function for the sequence $\{x_i\}$. By definition we have

$$z_i = z_{i-1} + x_i.$$

The corresponding auto-correlation function of the z -constrained sequence $\{x_i\}$ in terms of the auto-correlation function $R_z(k)$ is

$$R_x(k) = \begin{cases} 2R_z(k) - R_z(k-1) - R_z(k+1), & k \neq 0, \\ 1, & k = 0, \end{cases} \quad (8.17)$$

or, in frequency domain terms

$$H_x(\omega) = \frac{H_z(\omega)}{2(1 - \cos \omega)}, \quad (8.18)$$

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

We continue Example 8.1 with the computation of the power spectral density function of the maxentropic (z) sequence with digital sum variation $N = 5$, since it provides the opportunity to derive analytical expressions that can be evaluated easily without machine computations.

Example 8.2 The auto-correlation function is of the form

$$R_z(k) = \zeta^T \Pi Q^{|k|} \zeta.$$

The 5×5 state-transition matrix Q has five distinct eigenvalues $\{1, \mu_2, \dots, \mu_5\}$ and corresponding left eigenvectors $\mathbf{u}_1 = \boldsymbol{\pi}, \mathbf{u}_2, \dots, \mathbf{u}_5$. The eigenvectors are distinct and constitute a basis, so that

$$R_z(k) = \zeta^T \Pi Q^{|k|} \zeta = \left(\eta_1 \boldsymbol{\pi} + \sum_{j=2}^5 \mu_j^{|k|} \eta_j \mathbf{u}_j \right) \zeta. \quad (8.19)$$

The constants η_i , $i = 1, \dots, 5$, can be determined by evaluating $R_z(k)$, $k = 0, \dots, 4$. The eigenvalues of Q are $\mu_1 = 1$, $\mu_2 = -1$, $\mu_3 = 0$, $\mu_4 = 1/\sqrt{3}$, and $\mu_5 = -1/\sqrt{3}$. The corresponding left eigenvectors are $(1/12, 1/4, 1/3, 1/4, 1/12)$, $(1/12, -1/4, 1/3, -1/4, 1/12)$, $(-1/2, 0, 1, 0, -1/2)$, $(1, \sqrt{3}, 0, -\sqrt{3}, -1)$, and $(-1, \sqrt{3}, 0, -\sqrt{3}, 1)$. Since, for reasons of symmetry

$$\zeta(\sigma_i) = -\zeta(\sigma_{N-i+1}),$$

we obtain

$$\begin{aligned} R_z(k) &= \zeta^T \Pi Q^{|k|} \zeta = \mu_4^{|k|} \eta_4 \mathbf{u}_4 \zeta + \mu_5^{|k|} \eta_5 \mathbf{u}_5 \zeta \\ &= \alpha_1 \left(\frac{1}{3} \right)^{|k/2|} + \alpha_2 (-1)^{|k|} \left(\frac{1}{3} \right)^{|k/2|}. \end{aligned}$$

The constants α_1 and α_2 are found by substituting $R_z(0) = 7/6$ and $R_z(1) = 2/3$, from which we obtain

$$\alpha_1 = 7/12 + 1/\sqrt{3} \quad \text{and} \quad \alpha_2 = 7/12 - 1/\sqrt{3}.$$

After a straightforward calculation, we find for the auto-correlation function for the (z) sequence that takes $N = 5$ sum values:

$$R_x(k) = \begin{cases} 1, & k = 0 \\ -3^{-(k/2+1)}, & k > 0, k \text{ even} \\ -\frac{2}{3} 3^{-(k+1)/2}, & k > 0, k \text{ odd.} \end{cases} \quad (8.20)$$

Writing out the relation of the power spectral density $H_x(\omega)$ gives

$$\begin{aligned} H_x(\omega) &= 1 - \frac{2}{3} \operatorname{Re} \sum_{k=1}^{\infty} 3^{-k} e^{-j2k\omega} - \frac{4}{3} \operatorname{Re} \sum_{k=1}^{\infty} 3^{-k} e^{-j(2k-1)\omega} \\ &= \frac{4}{3} \frac{4 - \cos \omega - 3 \cos 2\omega}{5 - 3 \cos 2\omega}. \end{aligned}$$

Other closed-form expressions for the spectra of maxentropic (z) sequences were derived by Kerpez [200]. By way of illustration, we have plotted in Figure 8.3 the power spectral density function of maxentropic z -constrained sequences for various values of the digital sum variation N .

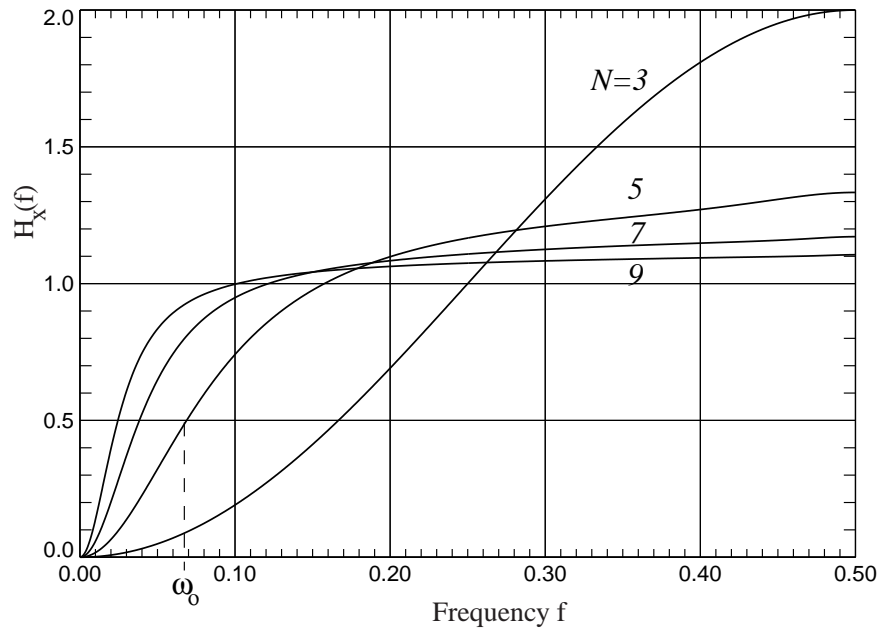


Figure 8.3: Power density function $H_x(\omega)$ of maxentropic (z) sequences against frequency $f = \omega/(2\pi)$ with digital sum variation N as a parameter. For the case $N = 5$, we have indicated the cut-off frequency ω_0 . We may read from the diagram that $\omega_0 \approx 2\pi \times 0.068 \approx 0.43$.

It is seen that, of course, the spectral density is zero at zero frequency, but it is more relevant to note that there is a region of frequencies, close to the zero frequency, where the spectral density is low. The width of this region, termed the *notch width*, is of great engineering relevance. Note in Figure 8.3 that the notch width decreases with mounting digital sum variation N . The appropriate engineering balance between system cost, in terms of redundancy, and performance, in terms of the width of the spectral notch, cannot be judged without some quantitative measure. The width of the spectral notch can be quantified by a parameter called the *cut-off frequency*. To this end, let $H(\omega)$ denote the power density function of a sequence with vanishing power at dc. The cut-off frequency of a dc-constrained sequence,

denoted by ω_0 , is formally defined by (see Figures 8.3 and 8.4, page 210), [177, 97]

$$H(\omega_0) = \frac{1}{2}. \quad (8.21)$$

In the next section, we will study the relationship between notch width and capacity.

8.3 Performance assessment

At this juncture we have completed our discussion of maxentropic dc-balanced sequences. Our next task is to determine some simple bounds to the performance of maxentropic dc-balanced sequences in terms of the spectral notch width, ω_0 , that is the range of frequencies with suppressed components of sequences with a spectral null at dc. These bounds underlie our subsequent analysis of the performance of implemented dc-balanced codes. We begin, since it is, as we will show shortly, closely related to the spectral notch width, by computing the sum variance of maxentropic dc-balanced sequences. The performance evaluation of various properties of dc-balanced sequences is based on work first presented by Justesen [177]. He carried out a comprehensive study on rates and spectra of digital codes, and as a result of this work he discovered a remarkable and useful relation between the sum variance $s_z^2 = E\{z_i^2\}$ and the width of the spectral notch, ω_0 .

The variance of the running digital sum can be expressed concisely in terms of the auto-correlation function $R_x(i)$ of the dc-balanced sequence. To this end, consider

$$z_m - z_0 = x_1 + x_2 + \cdots + x_m.$$

The variance of the variable $z_m - z_0$ is

$$\begin{aligned} 2s_z^2 - 2E\{z_i z_{i+m}\} &= \sum_{j=-m+1}^{m-1} (m - |j|) R_x(j) \\ &= m \sum_{j=-m+1}^{m-1} R_x(j) - 2 \sum_{j=1}^{m-1} j R_x(j). \end{aligned}$$

Let $H_x(\omega)$ denote the power spectral density function of the sequence. Assuming the function $H_x(\omega)$ is more or less 'well behaved', that is, it is smooth and $H_x(\omega) \simeq a\omega^2$, $\omega \ll 1$, we take the limit for $m \rightarrow \infty$, and use

$$\lim_{m \rightarrow \infty} E\{z_i z_{i+m}\} = 0$$

and

$$\lim_{m \rightarrow \infty} \sum_{j=-m+1}^{m-1} R_x(j) = H_x(0) = 0,$$

we get

$$s_z^2 = - \sum_{i=1}^{\infty} i R_x(i). \quad (8.22)$$

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

$$\begin{aligned} R_x(0) &= 1, \\ R_x(i) &= \rho r^{|i|}, \quad i \neq 0, \quad |r| < 1, \end{aligned} \quad (8.23)$$

where the constant ρ is chosen in order to allow the spectrum to vanish at zero frequency, or $H_x^r(0) = 0$. Thus

$$H_x^r(0) = \sum_{i=-\infty}^{\infty} R_x(i) = 1 + 2 \sum_{i=1}^{\infty} R_x(i) = 0, \quad (8.24)$$

whence

$$\rho = -\frac{1}{2} \frac{1-r}{r}. \quad (8.25)$$

If $r = 0$ we have the special case for which

$$\begin{aligned} R_x(-1) &= R_x(1) = -1/2, \\ R_x(i) &= 0, \quad |i| > 1. \end{aligned}$$

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

$$\begin{aligned} H_x^r(\omega) &= \sum_{i=-\infty}^{\infty} R_x(i) e^{-j i \omega} \\ &= 1 + \rho \{ r e^{-j \omega} + r^2 e^{-j 2 \omega} + \dots + r e^{j \omega} + r^2 e^{j 2 \omega} + \dots \}. \end{aligned} \quad (8.26)$$

Now, since

$$\begin{aligned} \sum_{k=0}^{\infty} x^k &= \frac{1}{1-x}, \\ H_x^r(\omega) &= \rho \sum_{i=0}^{\infty} (r e^{-j \omega})^i + \rho \sum_{i=0}^{\infty} (r e^{j \omega})^i - 2\rho + 1 \\ &= (1+r) \frac{1 - \cos \omega}{1 + r^2 - 2r \cos \omega}. \end{aligned} \quad (8.27)$$

Figure 8.4 shows the power spectral density function $H_x^r(\omega)$ against the frequency $f = \omega/2\pi$ with the quantity r as a parameter. The subsequent analysis provides an important relationship between the cut-off frequency, the parameter r , and the sum variance. Substitution of $H_x^r(\omega)$ into (8.21) yields

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

or

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

For small values of ω_0 we may use the approximation $\cos \omega_0 \simeq 1 - \omega_0^2/2$, so that

$$\omega_0 \simeq 1 - r. \quad (8.28)$$

The use of (8.22) yields the following expression for the sum variance:

$$s_z^2 = - \sum_{i=1}^{\infty} i R_x(i) = \frac{1}{2(1-r)}. \quad (8.29)$$

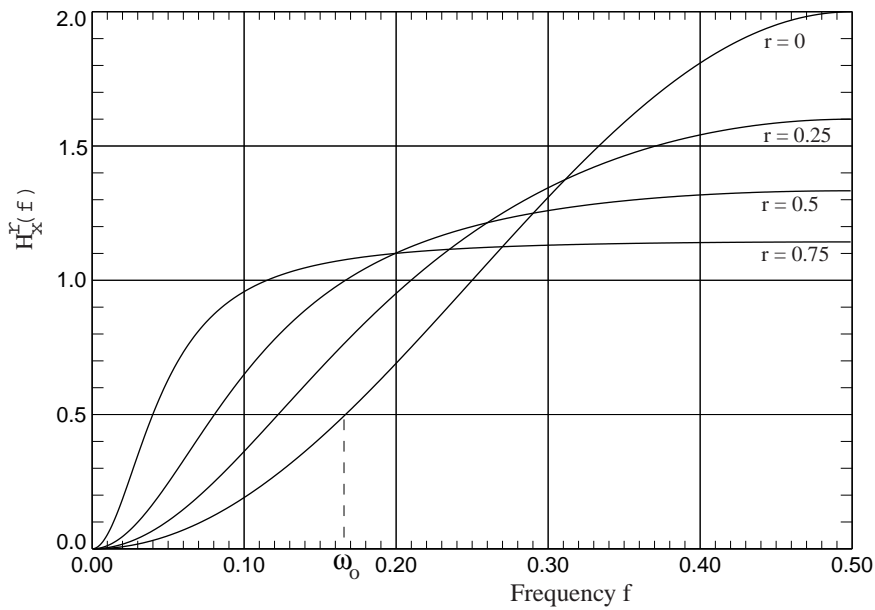


Figure 8.4: Power spectral density function $H_x^r(\omega)$ against frequency $f = \omega/(2\pi)$ of sequences with an exponential decay of its auto-correlation function $R_x(i) = \rho r^{|i|}$ with the quantity r as a parameter. By way of example, the cut-off frequency ω_0 is shown for the curve with parameter $r = 0$. It can be seen that a negative value of the parameter r shifts the power towards the upper end of the base band.

A combination (8.28) and (8.29), and a little rearrangement yields an approximation of the cut-off frequency ω_0 in terms of the sum variance, namely

$$2s_z^2\omega_0 \simeq 1. \quad (8.30)$$

Thus, for the sequence with an exponentially decaying auto-correlation function, we conclude that the cut-off frequency is approximately inversely

proportional to the variance of the running digital sum. When the auto-correlation function does not obey the assumed exponential decay, the situation is more complicated and no general statement can be made. In spite of the fact that the relationship between sum variance and cut-off frequency is only correct for dc-balanced sequences with an exponentially decaying auto-correlation, recent studies have shown that it also applies to sequences generated by implemented channel codes. Extensive computations of samples of implemented channel codes, made by Justesen [177] to validate the reciprocal relation (8.30) between cut-off frequency and sum variance, have revealed that this relationship is fairly reliable. This result has also motivated other researchers [353] to apply the sum variance as a valuable criterion of the low-frequency characteristics of a channel code, which is of practical significance since the sum variance of a dc-free sequence can often be evaluated by simple calculations, even though the auto-correlation function and corresponding spectrum are complicated functions.

Now that we have sharpened our tools with the above analysis, we can proceed with the performance analysis of maxentropic dc-free sequences. The sum variance $E\{z_i^2\}$ of the ensemble of maxentropic (z) sequences, denoted by $\sigma_z^2(N)$, where N is the DSV of the sequences, is simply given by

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

Using (8.13) and working out yields

$$\sigma_z^2(N) = \frac{2}{N+1} \sum_{k=1}^N \left(\frac{N+1}{2} - k \right)^2 \sin^2 \frac{\pi k}{N+1}.$$

It has been shown in a private communication by A.J. Janssen that the above summation can be succinctly rewritten as

$$\sigma_z^2(N) = \frac{1}{12}(N+1)^2 - \frac{1}{2 \sin^2 \frac{\pi}{N+1}} + \frac{1}{6}. \quad (8.31)$$

Results of computations are collected in Table 8.1. From Figure 8.3 we may read that, in the case $N = 5$, the indicated cut-off frequency is approximately 0.43. Given this, plus the fact that the sum variance $\sigma_z^2(5) = 7/6$, (see Table 8.1) we find that twice the product of sum variance and cut-off frequency is 1.008 which, at least in this specific case, is close to the value predicted by (8.30). A plot of the sum variance versus the redundancy $1 - C(N)$, shown in Figure 8.5, affords more insight into the trade-off between rate and performance of dc-balanced codes.

Figure 8.5 reveals that the relationship between the logarithms of the sum variance and the redundancy is approximately linear. The immediate

engineering implications of the preceding outcomes are quite relevant. In a Compact Disc player, for example, a high-pass filter is used in the receiver (player) to pass on the one hand the information signal and to reject, on the other hand, the low-frequency noise due to fingerprints on the disc. The high-pass filter is chosen as a sound trade-off between the two conflicting goals. The above theory reveals that more rejection of low-frequency noise components is possible only at the expense of the rate of the code used, which is proportional to the playing time of the disc. The type of curve shown in Figure 8.5 presents the designer with a spectral budget, that is, if the designer desires a certain width of the spectral notch, he/she knows the price in terms of code redundancy.

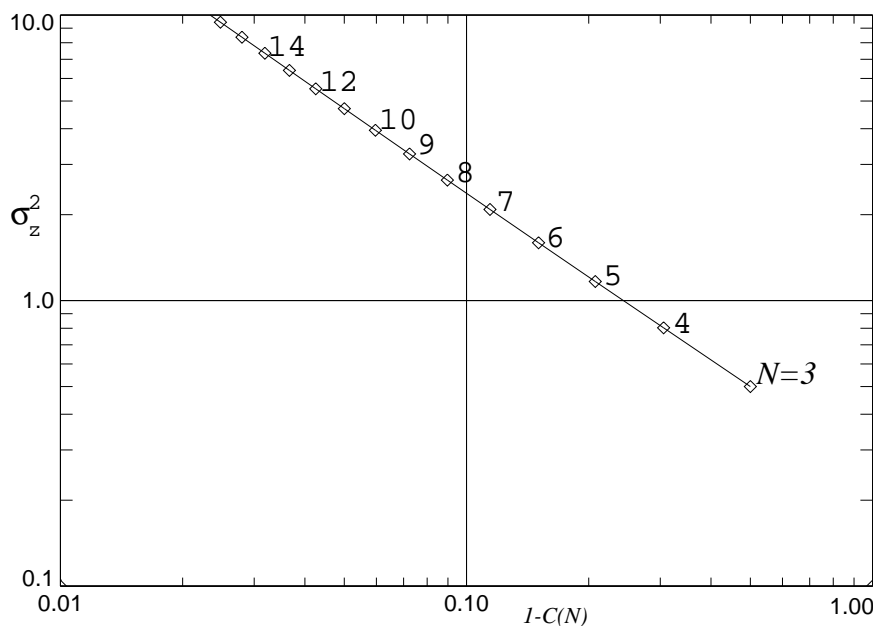


Figure 8.5: Sum variance versus redundancy of maxentropic (z) sequences. The relationship is plotted as a solid line. Note, however, that only a discrete set of points is achievable .

It is worth noting that the graphical results of Figure 8.5 can be expressed analytically for large values of the digital sum variation N . From (8.31) it is immediate that

$$\sigma_z^2(N) \simeq \left(\frac{1}{12} - \frac{1}{2\pi^2}\right)(N+1)^2 + O\left(\frac{1}{(N+1)^2}\right). \quad (8.32)$$

Similarly

$$C(N) \simeq 1 - \frac{\pi^2}{2 \ln 2} \frac{1}{(N+1)^2}, \quad N \gg 1. \quad (8.33)$$

The above approximations plus inspection of (8.12) and (8.31) lead to a fundamental relation between the redundancy $1 - C(N)$ and the sum variance of a maxentropic (z) sequence, namely

$$0.25 \geq (1 - C(N))\sigma_z^2(N) > \frac{\pi^2/6 - 1}{4 \ln 2} = 0.2326. \quad (8.34)$$

Actually, the right-hand bound is within 1% accuracy for $N > 9$. Equation (8.34) states that, for large enough N , the product of redundancy and sum variance of maxentropic RDS-constrained sequences is approximately constant. The relationship between sum variance (and implicitly, according to (8.30), the cut-off frequency) and redundancy is very interesting, since it reflects what we intuitively expect and illustrates very well the phrase that there is no such thing as a free lunch. For a wider frequency range of suppressed components, one has to pay more in terms of redundancy of the sequence. Equation (8.34) will be exploited in the subsequent chapter to establish a general figure of merit of implemented dc-constrained codes. In summarizing the lessons to be learnt from the theory, it should be borne in mind that the treatment given here applies solely to RDS-constrained sequences. If the sequences comply with other constraints as well, for example runlength constraints then, in general, the results derived above lose their validity.

8.4 Simple coding schemes

In the next subsections, we will describe in more detail coding schemes for generating dc-free sequences.

8.4.1 Zero-disparity coding schemes

Let a codeword \mathbf{x} of length n , n even, consist of bipolar symbols x_i , $1 \leq i \leq n$, $x_i \in \{-1, 1\}$. The disparity $d(\mathbf{x})$ of a codeword is the sum of the codeword symbols, or

$$d(\mathbf{x}) = \sum_{i=1}^n x_i. \quad (8.35)$$

An obvious basic code to generate dc-free sequences, and certainly the simplest to describe, is constituted by zero-disparity codewords, i.e. of codewords \mathbf{x} for which $d(\mathbf{x}) = 0$. In this scheme, each source word is uniquely represented by a codeword that contains equally many 'one's and 'zero's. The number of zero-disparity codewords, N_0 , of binary symbols (n even) is given by the binomial coefficient

$$N_0 = \binom{n}{n/2}. \quad (8.36)$$

Table 8.2 shows the number of zero-disparity codewords as a function of the codeword length n . Table 8.2 also presents the code rate, R , given by

$$R = \frac{1}{n} \log_2 N_0. \quad (8.37)$$

The zero-disparity codewords can be concatenated without a merging rule. That is, the sequence is encoded without information about the history, and, obviously, there is a fixed relationship between codewords and source words. Practical coding arrangements demand that the number of codewords is a power of 2, so that a subset of the N_0 available codewords must be used, thus effectively lowering the code rate R . For example, since $2^{17} = 131,072$ and $N_0 = 184,756$, we could take $m = 17$ and $n = 20$.

Table 8.2: Number of zero-disparity codewords and code rate versus codeword length n .

n	N_0	R
2	2	0.500
4	6	0.646
6	20	0.720
8	70	0.766
10	252	0.798
12	924	0.821
14	3432	0.839
16	12870	0.853
18	48620	0.865
20	184756	0.875

For large values of n the binomial coefficient (8.36) can be approximated using Stirling's formula [206]. Then we find for the code rate

$$R \approx 1 - \frac{1}{2n} \log_2 n, \quad n \gg 1. \quad (8.38)$$

The translation of source words into zero-disparity codewords is a challenging task, in particular when the codeword length is long. As discussed in Example 6.2, page 139, enumerative encoding, using Pascal's triangle, is an attractive method, whose roots in the literature go back to the early 50s [216]. Alternatively the method by Knuth and Henry, to be discussed in Section 8.9, can be used.

8.4.2 Low-disparity coding schemes

Up to this point we have been concerned with the category of dc-balanced codes which exclusively uses codewords of zero-disparity. A generalization of the zero-disparity coding principle, quite straightforwardly, leads to the alternate or low-disparity coding technique. Besides the set of zero-disparity codewords, sets of codewords with non-zero disparity are used. The archetypal code has two alternative representations of the source words. The two alternative channel representations have opposite disparity. The choice of representing the source word with a codeword of positive or negative disparity is stipulated by the polarity of the running digital sum just before transmission of the new codeword. The encoder opts for a particular channel representation with the aim of minimizing the absolute value of the running digital sum after transmission of the new codeword. Zero-disparity codewords can in principle be exploited in both modes and are usually uniquely allocated to source words. For ease of implementation, zero-disparity codewords are sometimes divided into two sets that are used in both modes.

The assignment of the source words to the various codewords is to some extent arbitrary. There is, however, one important requirement that has to be taken into account: the decoding should be state-independent to circumvent serious error propagation. The structure of the low-disparity encoder makes it possible, in all the most common practical codes, to choose an assignment that permits state-independent decoding.

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

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

$$N_j = \binom{n}{n/2 + j}, \quad 0 \leq j \leq K. \quad (8.39)$$

The total available number of codewords in S_+ (or for symmetry reasons in

S_-), denoted by M , is

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

The code rate is simply given by

$$R = \frac{1}{n} \log_2 M. \quad (8.40)$$

An essential part of the low-disparity encoder is a counter which registers the running digital sum. As the choice between a positive or negative disparity of the codewords is made on a per word basis so as to minimize the absolute value of the running digital sum after transmission of the new codeword, it is not difficult to see that the running digital sum takes on a finite number of values during transmission. Without loss of generality it can be assumed (by a proper setting of the initial sum value at the beginning of the transmission) that the sum values are symmetrically centered around zero. The set of values the RDS assumes at the end (or start) of a codeword can be associated with encoder states termed the *terminal* or *principal* states. It is immediate that the set of principal states is a subset of the set of all RDS values the sequence can take.

Let the terminal digital sum after transmission of the t -th transmitted codeword be $\Delta^{(t)}$. The sum after transmission of the $(t+1)$ -th codeword is

$$\Delta^{(t+1)} = \Delta^{(t)} + d^{(t+1)},$$

where $d^{(t+1)}$ is the disparity of the $(t+1)$ -th codeword. The sign of the disparity, by appropriately selecting the channel representation from S_+ or S_- , is chosen to minimize the accumulated sum $\Delta^{(t+1)}$. A code based on this algorithm is said to be balanced. After some thought we conclude that the quantity $\Delta^{(t)}$ may assume one of the $2K$ values from the set $\mp 1, \mp 3, \dots, \mp(2K-1)$, provided the encoder is properly initialized. Apparently, an encoder ruled by the previous algorithm generates a sequence whose accumulated digital sum is preserved within some limits. It can easily be verified that the total number of RDS values that the sequence can take, i.e. the digital sum variation, is

$$N = 2(2K - 1 + \frac{n}{2}) + 1 = 4K + n - 1. \quad (8.41)$$

As an illustration, the RDS is shown as a function of symbol time interval in Figure 8.6. The lines display the running digital sum of each of the codewords and so give a visual picture of the encoding process. A chart like this is called a *trellis diagram*. The code has codeword length of $n = 6$ and it is constituted by the maximum number $K+1 = n/2+1 = 4$ subsets. Note

the $2K = 6$ allowed sum values at the end of each codeword and also the $N = 4K + n - 1 = 17$ values that the running digital sum can take within a codeword. The encoder which generates a low-disparity sequence can be modelled as a finite-state machine. The conjunction of a source word and the terminal sum value $\Delta^{(t)}$ determines the actual transmitted codeword and the next terminal sum value $\Delta^{(t+1)}$. Thus, the set of encoder states, (or principal states) designated by $\Sigma_p = \{\sigma_1, \dots, \sigma_{2K}\}$, is the set of terminal sum values.

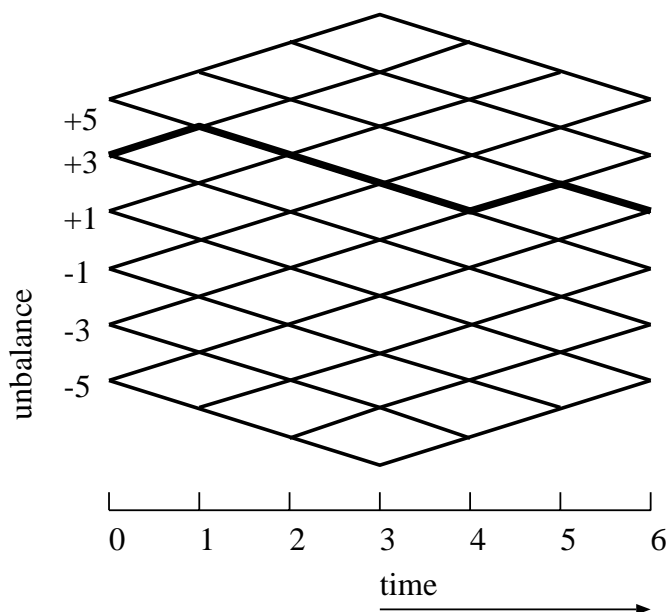


Figure 8.6: Unbalance trellis diagram. The lines display the running digital sum of each permitted codeword and give a visual picture of the encoding process. The code has codeword length $n = 6$ and it uses the four subsets. The thick curve shows the path taken by the codeword '+ - - - + -', assuming that its initial unbalance is +3. The diagram is similar to Figure 8.1, page 199, but here the time axis is initialized at the start of each codeword.

In the computation of the power density function and the sum variance of the encoded stream, we need to know the stationary probability of being in a certain terminal state. For this purpose, we assume the source words to be generated by a random independent process. Then the signal process $\Delta^{(t)}$ is a simple stationary Markov process. The value that $\Delta^{(t)}$ can take is related to one of the $2K$ states of the Markov process (see Chapter 3). The state-transition probability matrix Q whose entries are represented by q_{ij} , where q_{ij} is the probability that the next codeword will take it to terminal state σ_j given that the encoder is currently in state σ_i , can easily be found.

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

terminal states

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

The transition probability q_{ij} is the proportion of codewords in the mode used in state σ_i having the appropriate disparity d for the transition to encoder state $\sigma_j = \sigma_{i+d/2}$. The transition probability p_i equals the relative number of codewords in subset S_i , or $p_i = N_i/M$. The special structure of the matrix Q allows us to establish a closed expression of the steady-state probability vector $\boldsymbol{\pi}$ whose entries are, for reasons of symmetry, $(\pi_K, \dots, \pi_1, \pi_1, \dots, \pi_K)$, where π_i is the probability of being in the encoder state σ_i with corresponding sum value $(2i - 1)$. By definition we have the identity

$$\boldsymbol{\pi}Q = \boldsymbol{\pi},$$

which amounts to the following system of linear equations:

$$\pi_K p_j + \pi_{K-1} p_{j-1} \dots + \pi_1 p_{K-j} + \pi_2 p_{K-j+1} \dots = \pi_{K-j}, \quad j = 0, \dots, K-1.$$

After evaluating we arrive at

$$\rho \pi_{K-i} = \sum_{j=K-i}^K p_j, \quad 0 \leq i \leq K-1, \quad (8.42)$$

where ρ is determined from the normalization

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

With (8.42), we find

$$\rho = 2 \sum_{i=1}^K i p_i. \quad (8.43)$$

8.4.3 Polarity switch method

Bowers [42] and Carter [50], published a construction of dc-balanced codes where the $(n - 1)$ source symbols are supplemented by one extra bit set to 'one' to form an n -tuple. The code rate of the polarity bit code is

$$R = 1 - \frac{1}{n}.$$

If the accumulated disparity at the start of the transmission of a new n -tuple and the disparity of the new n -tuple have the same sign, then all symbols

in the n -tuple (i.e. including the polarity bit) are inverted (complemented), otherwise the n -tuples are left intact before transmission. The accumulated disparity is tallied by a reversible counter connected to the output line. If the disparity of the codeword is zero, the encoder has more room for making an optimal choice. Spectral line components can be avoided by randomly setting the polarity bit. Greenstein [116] suggested to transmit the words such that the 'polarity' of the words equals that of its most recent nonzero value. Alternatively, Kim [205] proposed to invert a subset of the codeword symbols. A performance analysis is not available in the literature. Murdock [261] presented an embellishment of the basic scheme.

The decoder is very simple, since it merely observes a possible inversion of a received n -tuple by inspecting the sign of the polarity bit and, according as this is positive or negative, reads the remaining bits directly or complemented.

8.5 Computation of the spectrum

The codeword symbols are, in general, transmitted serially at intervals of unit duration. The emitted signal $x(t)$ can be expressed by

$$x(t) = \sum_{j=-\infty}^{\infty} \sum_{i=1}^n x_{j,i} s[t - (jn + i - 1)], \quad (8.44)$$

where we revert to the notation of Chapter 3, and denote the j th transmitted codeword in the sequence by a row vector \mathbf{x}_j with elements $x_{j,i}$. The symbols $x_{j,i}$ are governed by the code rules and the source words to be encoded.

In Chapter 3, a complete and systematic analysis for computing the auto-correlation function of fixed-length-codeword-based channel codes has been provided. The analysis presented can, in principle, be directly used to compute the spectrum of alternate bi-mode codes. The structure of full-set alternate bi-mode codes allows a more efficient computation to be detailed here. The main advantage of our approach is that we obtain a closed formula for the spectrum of codes with two subsets and simple expressions for code spectra with a larger number of subsets. The spectral density is computed assuming that the source words are equiprobable.

In general, the power spectral density function can be written as, see (3.42), page 40,

$$H_x(\omega) = H_{xc}(\omega) + H_{xd}(\omega) \sum_{k=-\infty}^{\infty} 2\pi\delta(\omega - 2\pi k/n), \quad (8.45)$$

where $H_{xc}(\omega)$ and $H_{xd}(\omega)$ represent the continuous and discrete components

of the spectrum. According to (3.43) and (3.44), page 41, we have

$$H_{xc}(\omega) = \frac{1}{n}\boldsymbol{\omega}(R_0 - R_\infty)\boldsymbol{\omega}^* + \frac{2}{n}\operatorname{Re}\sum_{k=1}^{\infty}\boldsymbol{\omega}(R_k - R_\infty)\boldsymbol{\omega}^*e^{-jkn\omega}. \quad (8.46)$$

and

$$H_{xd}(\omega) = \frac{1}{n^2}\boldsymbol{\omega}R_\infty\boldsymbol{\omega}^*, \quad (8.47)$$

where

$$R_k = E\{\mathbf{x}_l^T \mathbf{x}_{l+k}\}, \quad l \text{ any integer,}$$

and R_∞ is the limit of the correlation matrices R_k as $k \rightarrow \infty$. The row vector $\boldsymbol{\omega}$ with transposed conjugate $\boldsymbol{\omega}^*$ is defined as

$$\boldsymbol{\omega} = (e^{j\omega}, \dots, e^{jn\omega}). \quad (8.48)$$

The following two observations facilitate the computation of the spectrum and, in fact, they provide the grounds that allow us to write down a simple closed-form expression for the spectral contents of the dc-balanced sequence under consideration.¹

1. The source words are represented by zero-disparity codewords or are alternately represented by codewords of opposite polarity of the disparity. Since the source words are assumed to be drawn with equal probability, we conclude that codewords of disparity d and $-d$ are transmitted with equal probability. Thus we find $R_\infty = 0$ or, stated alternatively, it appears that the discrete components vanish in this case the spectrum of full-set low-disparity schemes is continuous.
2. The correlation between symbols of \mathbf{x}_{j_1} and \mathbf{x}_{j_2} , $j_1 \neq j_2$, depends only on the number of codewords separating the two codewords \mathbf{x}_{j_1} and \mathbf{x}_{j_2} , that is, on the number of codewords in the interval (j_1, j_2) . This can be seen as follows. Assume that the source symbols are produced by a random and independent process then, as a result, the consecutive codewords would also be uncorrelated. Any correlations are therefore due only to the alternations of the codewords. As an immediate consequence, the expectation $E\{\mathbf{x}_{j_1}^T \mathbf{x}_{j_2}\}$ depends, at most, on the number of codeword alternations between \mathbf{x}_{j_1} and \mathbf{x}_{j_2} .

Since, as said above, the correlation between symbols in \mathbf{x}_{j_1} and \mathbf{x}_{j_2} $j_1 \neq j_2$, depends only on the number of codewords in the interval (j_1, j_2) , we infer

¹The nice properties only hold provided all possible codewords of the various subsets are used. If the subsets are truncated, for example, to meet specific considerations or to cater for a convenient size of repertoire, such as a power of two, the subsequent analysis can only be used as an approximation.

that the correlation matrix $R_k = E\{\mathbf{x}_l^T \mathbf{x}_{l+k}\}$, $k = 1, 2, \dots$, consists of n^2 identical entries which we will denote by the scalar r_k , so that

$$R_k = r_k U, \quad k = 1, 2, \dots, \quad (8.49)$$

where U is an $n \times n$ matrix, each of whose elements is unity. The correlation matrix R_0 , on the other hand, has unity elements on the main diagonal; the other elements are all equal and are denoted by r_0 . Thus

$$R_0 = (1 - r_0)I + r_0 U, \quad (8.50)$$

where I denotes the identity matrix. According to (3.39), page 39, the phase-averaged auto-correlation function $R(\cdot)$ is

$$nR(i + jn) = \sum_{m=1}^{n-i} [R_j]_{m, i+m} + \sum_{m=n-i+1}^n [R_{j+1}]_{m, i+m-n}, \quad 0 \leq i \leq n-1 \quad (8.51)$$

where $[R_j]_{u,v}$ denotes the u, v entry of R_j . The auto-correlation function $R(\cdot)$ can now be expressed in terms of the set of numbers $\{r_i\}$:

$$\begin{aligned} R(i + jn) &= \frac{1}{n} \{(n-i)r_j + ir_{j+1}\}, \quad j \geq 0, \quad 0 \leq i \leq n-1, \\ R(0) &= 1. \end{aligned} \quad (8.52)$$

In words, the auto-correlation coefficients $R(i + jn)$ can be computed by a linear interpolation of the quantities r_j and r_{j+1} . The spectrum $H(\omega)$ (note that for the sake of convenience we omitted the subscript) of such a sequence is

$$H(\omega) = \frac{1}{n} \boldsymbol{\omega} R_0 \boldsymbol{\omega}^* + \frac{2}{n} \operatorname{Re} \sum_{k=1}^{\infty} \boldsymbol{\omega} R_k \boldsymbol{\omega}^* e^{-jkn\omega}.$$

This expression for the continuous spectrum contains matrix products which can be simplified. Using definition (8.48), we have

$$\boldsymbol{\omega} R_0 \boldsymbol{\omega}^* = n(1 - r_0) + r_0 \boldsymbol{\omega} U \boldsymbol{\omega}^*.$$

Using the above results yields

$$H(\omega) = 1 - r_0 + \frac{1}{n} r_0 \boldsymbol{\omega} U \boldsymbol{\omega}^* + \frac{2}{n} \boldsymbol{\omega} U \boldsymbol{\omega}^* \operatorname{Re} \sum_{k=1}^{\infty} r_k e^{-jkn\omega}.$$

The expression $\omega U \omega^*$ can be worked out as follows

$$\begin{aligned}
\omega U \omega^* &= [e^{j\omega} \ e^{j2\omega} \ \dots \ e^{jn\omega}] \begin{bmatrix} 1 & 1 & \dots & 1 \\ 1 & 1 & \dots & 1 \\ \cdot & \cdot & & \cdot \\ 1 & 1 & \dots & 1 \end{bmatrix} \begin{bmatrix} e^{-j\omega} \\ e^{-j2\omega} \\ \cdot \\ e^{-jn\omega} \end{bmatrix} \\
&= \sum_{i=1}^n \sum_{k=1}^n e^{j(i-k)\omega} \\
&= \sum_{k=-n+1}^{n-1} (n - |k|) e^{jk\omega} \\
&= n + 2 \sum_{k=1}^{n-1} (n - k) \cos k\omega = \left(\frac{\sin n\omega/2}{\sin \omega/2} \right)^2 \\
&= n^2 F^2(\omega),
\end{aligned}$$

where

$$F(\omega) = \frac{\sin n\omega/2}{n \sin \omega/2}.$$

After evaluation of the preceding expression, we obtain the spectrum in terms of the quantities r_i , namely

$$H(\omega) = 1 - r_0 + nF^2(\omega) \left\{ r_0 + 2 \sum_{i=1}^{\infty} r_i \cos in\omega \right\}. \quad (8.53)$$

The spectrum vanishes at $\omega = 0$, or $H(0) = 0$, so that we conclude

$$1 + (n - 1)r_0 + 2n \sum_{i=1}^{\infty} r_i = 0. \quad (8.54)$$

As a direct consequence, we derive for sequences that are a concatenation of zero-disparity blocks, which obviously satisfy $r_i = 0$, $i \neq 0$,

$$r_0 = -\frac{1}{n - 1}.$$

The corresponding auto-correlation function $R(k)$ of the concatenated sequence follows from (8.52):

$$R(k) = \begin{cases} 1, & k = 0 \\ \frac{k - n}{n(n - 1)} & 0 < k \leq n \\ 0, & k > n. \end{cases} \quad (8.55)$$

The power density function of zero-disparity codeword-based channel codes is

$$H(\omega) = \frac{n}{n-1} \{1 - F^2(\omega)\}, \quad (8.56)$$

which agrees with the outcomes derived in Example 3.4, page 47.

The calculation of the numbers r_i , $K > 0$, is a tedious, but straightforward evaluation of (8.46) and the results of Chapter 3. Therefore we merely state the final results. The correlation function r_i is given by

$$r_i = \mathbf{c}_1^T Q^{i-1} \mathbf{c}_2, \quad i \geq 1, \quad (8.57)$$

where Q is the state-transition probability matrix and

$$Q^0 = I.$$

The $2K$ -vectors \mathbf{c}_1 and \mathbf{c}_2 are given in the interval $1 \leq i \leq K$ by

$$c_1(i+K) = -\frac{2}{n} \left\{ \sum_{j=i+1}^K (j-i)\pi_j p_{j-i} - \sum_{j=0}^{K-i} (j+i)\pi_{j+1} p_{j+i} \right\}$$

and

$$c_2(i+K) = -\frac{2}{n} \sum_{j=1}^K j p_j. \quad (8.58)$$

For symmetry reasons:

$$c_1(i) = -c_1(2K-i+1) \text{ and } c_2(i) = -c_2(2K-i+1), \quad 1 \leq i \leq K.$$

The correlation coefficient r_0 is not found by (8.57). The number r_0 equals the correlation of symbols in the same codeword or $r_0 = E\{x_{j_1} x_{j_2}\}$, where the symbol positions j_1 and j_2 are in the same codeword and $j_1 \neq j_2$. The coefficient r_0 can be computed with (8.54). A closed-form expression is derived in the Appendix to this chapter, page 239, where the sum variance of alternate codes is computed (see, for example, (8.83)).

Example 8.3 As an illustration, we study the simplest type of alternate code, in which two alternative codewords may represent a given source word. We will determine the spectrum and auto-correlation function for such a code in terms of the codeword length. To this end, consider the two-state process with $K = 1$. The preceding expressions for the spectrum and correlation function now become manageable. For reasons of symmetry, the steady-state distribution is $\boldsymbol{\pi} = (1/2, 1/2)$. The cardinalities of the two codeword subsets S_0 and S_1 are

$$N_0 = \binom{n}{n/2} \text{ and } N_1 = \binom{n}{n/2+1},$$

whence

$$p_0 = \frac{N_0}{N_0 + N_1} = \frac{n+2}{2(n+1)}$$

and

$$p_1 = 1 - p_0 = \frac{n}{2(n+1)}.$$

Substitution in (8.58) yields

$$c_1(1) = -c_1(2) = -\frac{p_1}{n} = \frac{-1}{2(n+1)}$$

and

$$c_2(1) = -c_2(2) = \frac{2p_1}{n} = \frac{1}{(n+1)}.$$

The 2×2 transition matrix Q is given by

$$Q = \begin{bmatrix} p_0 & p_1 \\ p_1 & p_0 \end{bmatrix} = \frac{1}{2(n+1)} \begin{bmatrix} n+2 & n \\ n & n+2 \end{bmatrix}.$$

An analytic expression for Q^i is

$$Q^i = \frac{1}{2} \begin{bmatrix} 1 & 1 \\ 1 & 1 \end{bmatrix} + \frac{1}{2(n+1)^i} \begin{bmatrix} 1 & -1 \\ -1 & 1 \end{bmatrix}, \quad i = 1, 2, \dots$$

After evaluation of (8.57), we obtain

$$r_i = \mathbf{c}_1^T Q^{i-1} \mathbf{c}_2 = -\frac{1}{(n+1)^{(i+1)}}, \quad i = 1, 2, \dots$$

Applying (8.54) it follows that this equation also holds for $i = 0$, whence

$$r_i = -\frac{1}{(n+1)^{(i+1)}}, \quad i = 0, 1, \dots$$

Substituting this equation in (8.53) and working out will express the spectrum of the two-state alternate code as

$$H(\omega) = (1-a) \left\{ 1 - \frac{a^2 n^2 F^2(\omega)}{1+a^2+2a \cos n\omega} \right\}, \quad (8.59)$$

where $a = -1/(n+1)$. Evaluating yields the second derivative of the spectrum at $\omega = 0$ as

$$H^{(2)}(0) = \frac{1}{6}(n+2)(n+11). \quad (8.60)$$

Figure 8.7 shows the power density function of two-state low-disparity codes against the frequency $f = \omega/2\pi$, and codeword length n as a parameter.

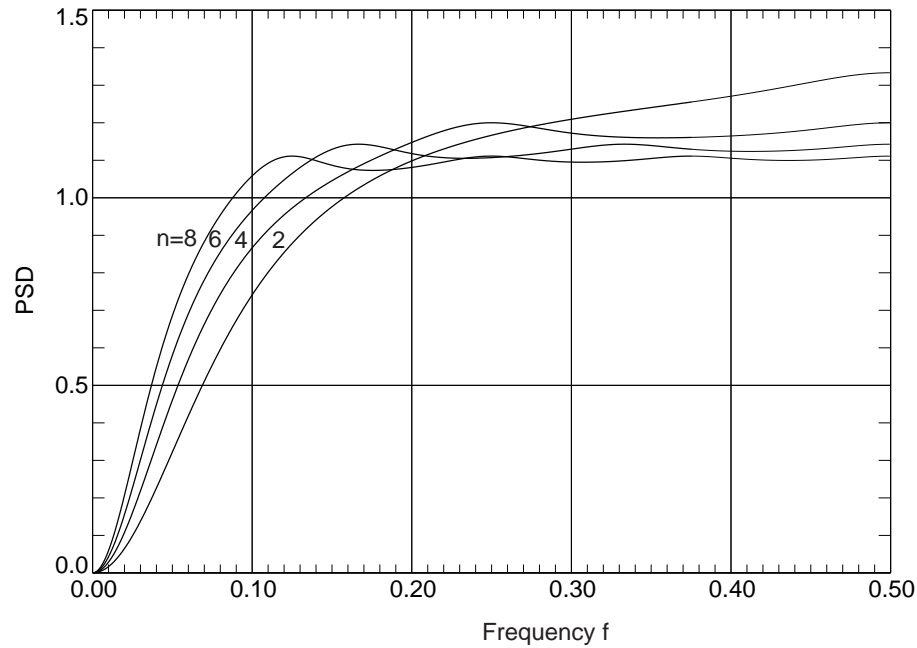


Figure 8.7: Power density function against frequency of low-disparity dc-free codes with two subsets versus frequency $f = \omega/2\pi$ and codeword length n as a parameter.

8.6 Performance appraisal

A notable frequency-domain characteristic of dc-balanced codes, the cut-off frequency, can be estimated by evaluating the sum variance of the code stream. The key to this approach is Justesen's relation (8.30), page 210,

$$2s_z^2\omega_0 \simeq 1,$$

which provides a simple relationship between the sum variance s_z^2 and the cut-off frequency ω_0 of a dc-balanced sequence. Thus the value of s_z^2 furnishes a straightforward characterization of the spectral behavior of dc-balanced codes. Of course, the reciprocal of the sum variance provides only an approximation to the cut-off frequency and although this weakness in our result is certainly undesirable, it appears to be a necessary compromise if we are to obtain simple analytical results. The advantage in simplicity to be gained from using the sum variance as a descriptor is obvious, since the only alternative now available for exactly computing the cut-off frequency is numerical analysis. The accuracy of Justesen's relation is a topic to be addressed in a later section.

Define the moments

$$u_m = \sum_{i=1}^K i^m p_i, \quad m \in \{1, 2, 3\}. \quad (8.61)$$

Then from the Appendix to this chapter, we find the variance of the terminal sum values

$$E\{z_0^2\} = 2 \sum_{i=1}^K (2i-1)^2 \pi_i = \frac{4}{3} u_3 - \frac{1}{3} \quad (8.62)$$

and the variance of the complete sequence

$$s_K^2 = \frac{4}{3} \frac{u_3}{u_1} + \frac{n-1}{6} - 2 \frac{n+1}{3n} u_2. \quad (8.63)$$

We now apply the results of the sum variance analysis in the following examples which describe easily calculable cases.

Example 8.4 When codewords of zero-disparity are employed exclusively, that is, $K = 0$, we obtain from (8.63), since $E\{z_0^2\} = 0$, the following expression for the sum variance:

$$s_0^2 = \frac{n+1}{6}.$$

Example 8.5 A closed-form expression for the sum variance can be obtained for codes where two subsets, namely S_0 and S_1 , are used for encoding. We invoke the numerical results derived in Example 8.3, page 223, and obtain

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

After substitution and working out (8.61) and (8.63) we obtain

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

so that

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

Example 8.6 The analysis developed in the preceding sections allows us to derive an expression for the sum variance of sequences generated by the polarity bit encoding principle. The code rate of the polarity bit code is

$$R = 1 - \frac{1}{n}.$$

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

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

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

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

The total number of codewords of non-negative disparity is

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

Using some properties of binomial coefficients, a routine computation yields

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

$$u_2 = \frac{1}{4}n$$

and

$$u_3 = \frac{n}{2}u_1.$$

Evaluation of (8.63) yields

$$s_P^2 = \frac{2n-1}{3}, \quad (8.64)$$

where s_P^2 designates the sum variance of the polarity bit encoded sequence.

Example 8.7 If all codewords are used, i.e. $K = n/2$, then

$$M = 2^{n-1} + \frac{1}{2} \binom{n}{n/2}$$

and

$$s_{n/2}^2 = \frac{5n-1}{6} - \frac{n+1}{12M}2^n. \quad (8.65)$$

For other values of the number of subsets used in the encoding table, it was not possible to obtain such simple analytical expressions. The outcomes of the computations are collected in Table 8.3, where the redundancy $1 - R$ and the digital sum variation N of the code are also given (see (8.41)). After a study of Table 8.1, page 203, and Table 8.3, several interesting facts now come to the fore. The remarkably simple block code with parameters $n = 2$ and $K = 0$, termed the *bi-phase* code, attains 100% of the capacity and the sum variance of the maxentropic sequence with digital sum variation $N = 3$.

Table 8.3: Sum variance, digital sum variation N and redundancy $1 - R$ of full-set alternate codes.

n	K	N	s_K^2	$1 - R$
2	0	3	0.50	.500
2	1	5	1.17	.208
4	1	7	1.50	.170
4	2	11	2.56	.135
6	1	9	1.83	.145
6	2	13	3.20	.107
6	3	17	3.94	.101
8	1	11	2.17	.128
8	2	15	3.68	.092
8	3	19	4.92	.083
8	4	23	5.32	.081

The two-state alternate block code with parameters $n = 2$ and $K = 1$ achieves 100% of the capacity and the sum variance of the maxentropic sequence with digital sum variation $N = 5$. It was shown by Ashley and Siegel [10] that other implemented 100% efficient binary dc-balanced codes are not possible. Figure 8.8 serves to provide a visual performance characterization of the implemented codes discussed above. Figure 8.8 shows the sum variance of various dc-balanced codes as a function of the redundancy $1 - R$ for selected values of the parameters K and n . For comparison purposes, the sum variance is plotted versus the redundancy $1 - C(N)$ of maxentropic (z) sequences (see (8.12) and (8.31)). There are some important conclusions that can be drawn. It is noteworthy that the performance of zero-disparity encoding diverges from the maxentropic bound with growing codeword length. It is also clear from the figure that the use of two codeword subsets, that is $K = 1$ is worthwhile in terms of sum variance and redundancy in a large range of the code redundancy.

For the sake of convenience we have opted to compute the sum variance in lieu of the cut-off frequency. The justification of this approach is based on Justesen's relation (8.30), page 210. To validate our course of action, the cut-off frequency was calculated by numerical procedures, invoking (8.53), (8.57), and (8.58), and compared with the reciprocal of the sum variance of the coded sequence. Consideration of computational results, in the range of code parameters listed in Table 8.3, reveals that Justesen's relation between sum variance and actual cut-off frequency is accurate to within a few percent, which is indeed a fascinating result.

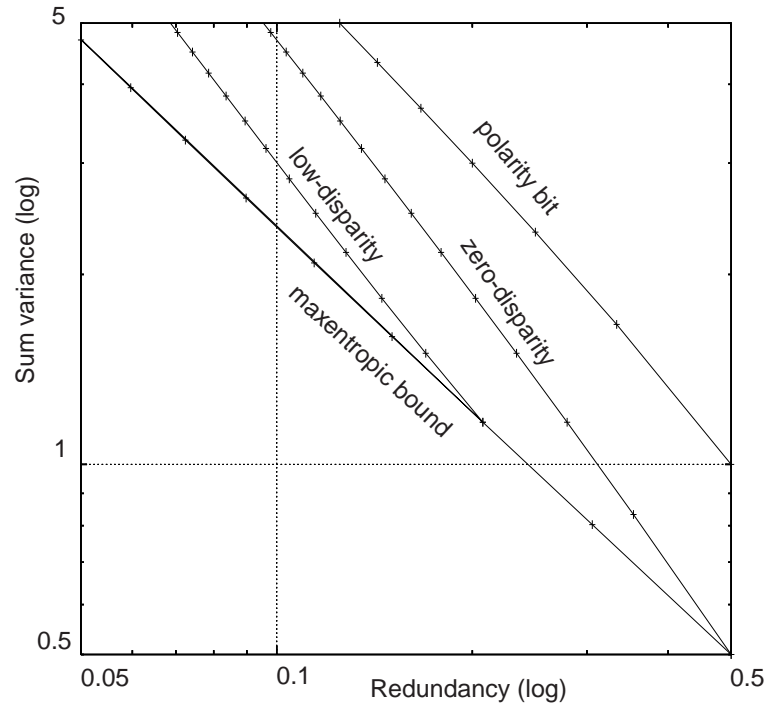


Figure 8.8: Sum variance of various dc-balanced codes as a function of the redundancy $1 - R$ with the code parameters K and n . As a reference, the sum variance is plotted versus the redundancy $1 - C(N)$ of maxentropic (z) sequences.

8.6.1 Efficiency of simple codes

We next employ the theory of maxentropic (z) sequences, developed in Section 8.2, to appraise the efficiency of the previously discussed dc-balanced codes. It is customary to define the *rate efficiency* of an implemented channel code as the ratio of the code rate and the noiseless channel capacity given the channel constraints, or

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

where η is the efficiency of the implemented code, $C(N)$ the capacity of the Chien channel (see (8.12), page 203) and N the digital sum variation of the channel code. In our context, the most desirable code is maxentropic and this will correspond with an efficiency of 100%.

As an illustration, let $n = 4$ and $K = 1$. In this instance, we find from Table 8.3 a sum variation and code rate $N = 7$ and $R = 0.83$, respectively, so that for this channel code an efficiency $\eta = 0.83/0.886 = 95\%$ (see Table 8.1, page 203) is concluded. The sum variance of the maxentropic sequence with

$N = 7$ is 2.09 (see again Table 8.1). The sum variance of the implemented code is 1.5, which amounts to $1.5/2.09 = 72\%$ of the sum variance of the maxentropic (z) sequence with $N = 7$. It is clear that the comparison of dc-balanced channel codes with maxentropic (z) sequences should take into account both the sum variance and the code rate. We thus define the encoder efficiency as

$$E = \frac{\{1 - C(N)\}\sigma_z^2(N)}{\{1 - R\}s^2}. \quad (8.66)$$

The encoder efficiency E , as defined in (8.66), compares the 'redundancy-sum variance products' of the implemented code and the maxentropic sequence with the same digital sum variation as the implemented code. Note that for $N > 9$ the 'redundancy-sum variance product' of maxentropic (z) sequences is approximately constant (see (8.34), page 213) and equals 0.2326.

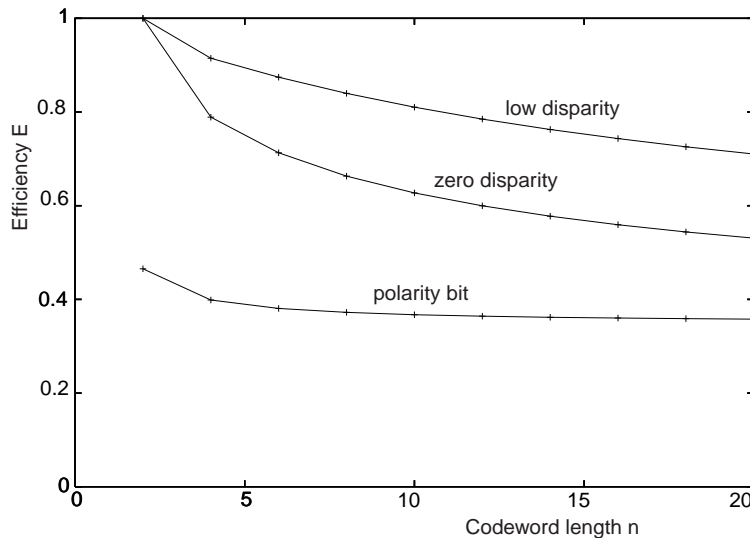


Figure 8.9: Efficiency of simple alternate channel codes.

The efficiency, E , of various codes versus codeword length is plotted in Figure 8.9. Examination of the family of result curves highlights several important characteristics. We observe that the low-disparity codes fall close to the performance of maxentropic sum constrained sequences, provided the codeword length is short. As the block length n increases, the efficiency of the implemented codes diminishes. For example, when $n = 2$ the efficiency of the zero-disparity code format is unity, it decreases to 60% when $n = 20$. Qualitatively, the same behavior is found for low-disparity coding schemes.

The polarity bit-encoding principle has the virtue of simple implementation, but on the other side of the balance, as we can see from Figures 8.8 and 8.9, it has a poor performance in terms of sum variance/redundancy

product and its performance is definitely far from optimal in the depicted range. The figures reveal that, for a given rate, the sum variance of a sequence encoded according to the polarity bit principle is typically 2.5 times that of maxentropic (z) sequences. Elaborating on (8.34), (8.63), and (8.66) shows that for long codewords the efficiency E asymptotically diminishes to 35%. In the range $0.8 < R < 0.9$, we note the debit side of this simple code: for a desired width of the spectral notch a loss of approximately 15% in code rate. Figures 8.8 and 8.9 show the superiority of zero-disparity codes with respect to polarity bit encoding in the most practical $(1 - R)$ interval. A calculation shows that for an unpractically long codeword of length $n > 160$, the polarity bit encoding principle outperforms the zero-disparity encoding.

8.7 High rate dc-balanced codes

In certain applications of channel codes, specifically digital recording on magnetic tape or transmission over a fiber-optic network, it has been found that a rate $R = 8/10$, dc-balanced channel code has attractive features [343, 342, 321] both in terms of system penalty and hardware realization. Most of the implementations in use are block codes which translate 1 byte (8 bits) into 10 channel symbols.

Clearly a zero-disparity block code is impossible since the number of available 10-bit zero-disparity codewords, 252, is smaller than required. A two-state encoder offers the freedom of at maximum $252 + 210 = 462$ codewords. Since only 256 codewords are required, this evidently offers a large variety of choices. The codebook can be tailored to particular needs, such as minimum dc-content and/or ease of implementation. It is therefore not too surprising that numerous variants have been described in the literature, in patent literature in particular. Table 8.4 shows the main parameters of selected dc-constrained 8b10b and 16b20b codes documented in the literature (for a more comprehensive survey of 8b10b codes, see the paper by Tazaki [321]).

The coding scheme designed by Widmer and Franaszek [342] is of interest due to its special structure. In their scheme, each incoming byte is partitioned into two sub-blocks of five and three bits, respectively. Five binary input lines are encoded into six binary output lines following the directions of the 5b6b look-up table and the disparity control. Similarly, the three remaining input bits are encoded into the remaining four output bits under the rules of a 3b4b look-up table and the disparity control. To a large extent the 5b6b and 3b4b encoders operate independently of each other on the basic principles of the bi-mode encoder. The number of available codewords is $(20 + 15) \times (6 + 4) = 350$. Modifications in the translation tables

have been made to reduce the maximum runlength and the digital sum variation, define special characters outside the data alphabet and minimize implementation cost at high data rates.

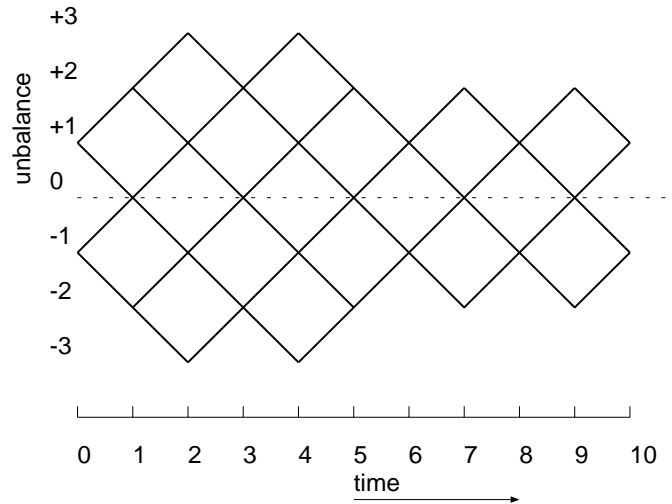


Figure 8.10: Unbalance trellis diagram of Widmer-Franaszek code. The 8b10b is constituted by a 5b6b and a 3b4b encoder which operate to a large extent independently in accordance with the basic principles of the bi-mode encoder. Note that the unbalance, or running digital sum, only takes two values at the symbol positions 0, 6, and 8.

A trellis diagram of the encoder is portrayed in Figure 8.10. An improvement of the above Widmer/Franaszek code was granted to Westby [339]. Another code of great practical interest is the Fukuda2 code, employed as the recording code in the DAT digital audio tape recorder [104]. The 8b10b code used in the DAT recorder is designed to function well in the presence of crosstalk from neighboring tracks, allow the use of the rotary transformer, and have a small ratio of maximum to minimum runlength in order to ease overwrite erasure. Essentially, the code operates on the low-disparity principle. The encoder has two states, and the codebook contains 153 zero-disparity codewords and 103 codewords of disparity ∓ 2 . The hardware of the encoder and decoder has been reduced by computer optimizing of the relationship between the 8-bit source words and the 10-bit codewords. This has been done so that the conversion can be performed in a small programmed logic array. The codewords are stored using NRZI notation. The details of the look-up tables for encoding and decoding can be found in Fukuda *et al.* [104]. Only codewords of disparity 0 and +2 are produced by the logic, since the codewords of disparity -2 can be obtained by reversing the first symbol of the codeword. The maximum runlength is 4 and the sum variance of this code is 1.71. A variant of this 8b10b code, also

reported by Fukuda *et al.*, possesses the virtue of reduced sum variance, 1.325, at the expense of a slightly increased maximum runlength.

Table 8.4: Main parameters of $R = 8/10$ dc-constrained codes.

<i>Reference</i>	N	T_{\max}
Stevens [309]	7	4
Shirota [299]	7	6
Widmer [343,342, 344]	7	5
Fukuda1 [104]	7	5
Fukuda2 [104]	7	4
Immink [147]	6	5
Fredrickson [99]	6	4

Very few attempts have been made to design a code with a rate greater than $8/10$. Specifically, codes of rate $16/18$ have been published. Widmer presented a rate $16/18$, $(0,6)$ dc-free code [341]. In a similar vein as the above rate $8/10$ code, the input data stream is broken into a 9-bit and a 7-bit sub-block and encoded separately into sub-blocks of 10 and 8 bits, respectively, while maintaining both dc balance and runlength constraints across all block and sub-block boundaries, i.e. the code comprises two alternating, bi-mode, codes of rate $9/10$ and $7/8$, respectively. A dc-free code of the same rate, $16/18$, was also presented by Soljanin [305]. The 16-bit input word is split into two bytes, i.e., into two 8-bit words, and each byte is mapped to a 9-bit word. Immink presented a rate $8/9$ code [153] that was found using the state-splitting design method (see Chapter 7). Other codes of odd codeword length will be discussed in the next section.

8.8 Dc-free code with odd codeword length

In this section, we will focus on dc-free codes with odd codeword length n .

Let \mathbf{x} denote the n -tuple (x_1, \dots, x_n) , where $x_i \in \{-1, 1\}$, then we define, see (8.35), the disparity of \mathbf{x} as

$$d(\mathbf{x}) = \sum_{i=1}^n x_i.$$

If we require that the running digital sum after concatenation of a new codeword is not larger (in absolute terms) than that at the beginning then each source word must have a representation of zero-disparity or it must have two alternative representations of opposite disparity. The words available

can easily be computed. Let N_- and N_+ denote the number of codewords with $d(\mathbf{x}) \leq 0$ and $d(\mathbf{x}) \geq 0$, respectively. Then, we find

$$N_- = N_+ = \begin{cases} 2^{n-1}, & n \text{ odd,} \\ 2^{n-1} + \frac{1}{2} \binom{n}{\frac{n}{2}}, & n \text{ even.} \end{cases} \quad (8.67)$$

When n is even, we are in the comfortable position that we can choose 2^{n-1} codewords from the many candidates available. For n odd, on the other hand, it can be seen that there are just enough codewords available to cater for a rate $(n-1)/n$ code. The implementation of the latter code is termed *polarity switch* code, where $(n-1)$ source symbols are supplemented by one symbol called the *polarity bit*. The encoder has the option to transmit the n -bit words "as is" or to invert all symbols. The choice of a specific translation is made in such a way that the RDS after transmission of the word is as close to zero as possible. The polarity bit is used by the decoder to undo the action of the encoder. Spectral properties of the polarity bit code have been investigated in the previous section. From the above deliberation, it seems, at first glance, that, for n odd, no codes other than the polarity switch method are feasible. In the next subsection, we will demonstrate otherwise.

8.8.1 Alternative definition of RDS.

The crux of the new class of dc-free code is the redefinition of the running digital sum. Figure 8.11 will be used to explain the various signals. The n -tuple \mathbf{y} is represented in NRZI notation, and translated into \mathbf{x} with symbols $\{1, -1\}$ using a change-of-state encoder (precoder). In the classical definition of the precoding operation, it is assumed that transitions occur at the start of the bit cells. In the new definition of the precoding operation, transitions are assumed to occur halfway the bit cells. It can be seen that the contribution to the RDS (after precoding using the new definition) of the 9-bit word $\mathbf{y} = '100100010'$ is nil, while, using the conventional precoding definition, see Figure 8.1, page 199, the contribution of the same input word \mathbf{y} is -1.

Let x_0 be the value of the last bit preceding the codeword \mathbf{x} . It can be verified that the contribution of the codeword \mathbf{y} to the RDS after precoding using the new definition, called the disparity of \mathbf{y} , and denoted by $ds(\mathbf{y}, x_0)$, is

$$ds(\mathbf{y}, x_0) = \sum_{i=1}^n x_i + (w(\mathbf{y}) \bmod 2)x_0, \quad (8.68)$$

where $w(\mathbf{y})$ denotes the weight of \mathbf{y} . The first right-hand term, $\sum x_i$, is the "conventional" contribution of \mathbf{x} to the RDS, and the second term expresses

a correction term, which equals x_0 if the number of 'one's of the codeword \mathbf{y} is odd. If the number of 'one's of \mathbf{y} is even, we do not need to 'correct' the disparity. The fascinating result of this new definition is that zero-disparity i.e. $ds(\mathbf{y}, x_0) = 0$, codewords, are possible for n odd.

$$\begin{array}{rcccccccccc} y_i & 1 & 0 & 0 & 1 & 0 & 0 & 0 & 1 & 0 \\ x_i & -1 & -1 & -1 & +1 & +1 & +1 & +1 & -1 & -1 \end{array}$$

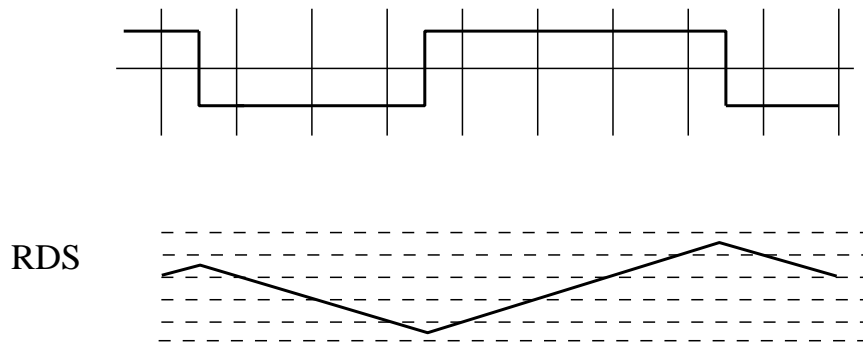


Figure 8.11: Running digital sum (RDS) versus time. The binary input symbols y_i are translated into the bipolar channel bits x_i using a precoder and a range converter. The transitions occur halfway the bit cells. See Figure 8.1, at page 199, for the classical definition of the RDS, where transitions occur at the beginning of each bit cell.

Let $N(s)$ denote the number of words \mathbf{y} having $ds(\mathbf{y}, 1) = s$. Then, it is not difficult to see that $N(s)$ satisfies

$$N(s) = \binom{n-1}{\frac{n-1}{2} + \lfloor \frac{s}{2} \rfloor}. \quad (8.69)$$

The zero-disparity words are uniquely allocated to the source words. Other codewords are allocated in pairs of opposite disparity. The choice of a specific representation is made to minimize the absolute value of the running digital sum. The words available in both modes can easily be computed. To that end, let \mathcal{N}_- and \mathcal{N}_+ denote the sets whose members \mathbf{y} satisfy $ds(\mathbf{y}, 1) \leq 0$ and $ds(\mathbf{y}, 1) \geq 0$, respectively, and let N_- and N_+ denote the cardinality of these sets. Then,

$$N_- = 2^{n-1} \quad (8.70)$$

and

$$N_+ = 2^{n-1} + \binom{n-1}{\frac{n-1}{2}}. \quad (8.71)$$

From (8.70) and (8.71) we infer that, as $N_- = 2^{n-1}$, the designer has no other choice than taking the words available, whereas, as $N_+ > 2^{n-1}$, the designer has the freedom to choose words that satisfy certain design criteria such as low coder/decoder complexity, minimizing maximum runlength, etc. It can easily be verified that both $10^{n-1} \in \mathcal{N}_-$ and $0^{(n-1)/2}10^{(n-1)/2} \in \mathcal{N}_-$, where 0^p denotes a string of p 'zero's. We conclude therefore that the maximum 'zero'-runlength equals $3(n-3)/2$.

The spectral performance of the new codes has been investigated by computer simulation. As an example we studied codes of rate 8/9. The number of codewords with non-negative disparity, N_+ , equals 326. We require 256 words, and from the codewords available we chose the words with the smallest absolute disparity. The maximum runlength is twelve.

Table 8.5: Sum variance of the new code, s^2 , and the sum variance, s_P^2 , of the polarity switch code versus codeword length n .

n	s^2	s_P^2
3	1.05	1.67
5	1.73	3.00
7	2.52	4.33
9	3.23	5.66
11	4.07	7.00

We computed the spectrum of the rate 8/9 code, and found that it has 4 dB less rejection at the low-frequency end than its 'polarity-bit' counterpart of the same rate. Also the maximum 'zero'-runlength is much larger, 18, than that of the new code. Table 8.5 lists the sum variance of the new code, s^2 , and the sum variance, s_P^2 , of the polarity switch code versus codeword length n . The sum variance, s_P^2 , has been computed by invoking (8.64), page 227. It can be seen that the new codes perform better as they show a smaller sum variance than that of the conventional scheme.

8.9 Balancing of codewords

The encoding technique discussed in the preceding sections is advantageous for implementation when the codewords are of medium size. An alternative and easily implementable encoding technique for zero-disparity codewords which is capable of handling (very) large blocks was described by Knuth

[206]. A patent was granted in 1982 to Henry describing a similar technique [131].

The method is based on the idea that there is a simple correspondence between the set of all m -bit binary source words and the set of all $(m+p)$ -bit codewords. The translation can in fact be achieved by selecting a bit position within the m -bit word which defines two segments, each having one half of the total block disparity. A zero-disparity block is now generated by the inversion of all the bits within one segment. The position information which defines the two segments is encoded in the p -bit prefix. The rate of the code is $m/(m+p)$.

Let $d(\mathbf{w})$ be the disparity of the binary source word $\mathbf{w} = (w_1, \dots, w_m)$, $w_i \in \{-1, 1\}$, or

$$d(\mathbf{w}) = \sum_{i=1}^m w_i. \quad (8.72)$$

Let $d_k(\mathbf{w})$ be the running digital sum of the first k , $k \leq m$, bits of \mathbf{w} , or

$$d_k(\mathbf{w}) = \sum_{i=1}^k w_i, \quad (8.73)$$

and let $\mathbf{w}^{(k)}$ be the word \mathbf{w} with its first k bits inverted. For example, if $\mathbf{w} = (-1, 1, 1, 1, -1, 1, -1, 1, 1, -1)$, we have $d(\mathbf{w}) = 2$ and $\mathbf{w}^{(4)} = (1, -1, -1, -1, -1, 1, -1, 1, 1, -1)$. If \mathbf{w} is of even length m , and if we let $\sigma_k(\mathbf{w})$ stand for $d(\mathbf{w}^{(k)})$, then the quantity $\sigma_k(\mathbf{w})$ is

$$\begin{aligned} \sigma_k(\mathbf{w}) &= -\sum_{i=1}^k w_i + \sum_{i=k+1}^m w_i \\ &= -2\sum_{i=1}^k w_i + d(\mathbf{w}). \end{aligned} \quad (8.74)$$

It is immediate that $\sigma_0(\mathbf{w}) = d(\mathbf{w})$, (no symbols inverted) and $\sigma_m(\mathbf{w}) = -d(\mathbf{w})$ (all symbols inverted). We may as $\sigma_{k+1}(w) = \sigma_k(w) \mp 2$, conclude that every word \mathbf{w} , m even, can be associated with at least one position k for which $\sigma_k(\mathbf{w}) = 0$, or $\mathbf{w}^{(k)}$ is balanced. The value of k is encoded in a (preferably) zero-disparity word \mathbf{u} of length p , p even. The maximum codeword length of \mathbf{w} is, since the prefix has an equal number of 'one's and 'zero's, governed by

$$m \leq \binom{p}{p/2}. \quad (8.75)$$

If m and p are both odd, we can use a similar construction. Modifications of the generic scheme are discussed in Knuth [206], Alon *et al.* [6], Al-Bassam & Bose [23], and Tallini, Capocelli & Bose [314]. The rate of the above construction is

$$R \approx 1 - \frac{1}{n} \log_2 n, \quad n \gg 1, \quad (8.76)$$

which shows that the redundancy is a factor of two larger than that of the full set of zero-disparity codes, see (8.38). Note that if the codewords are given in NRZI notation, that the zero-disparity codewords can be obtained by inverting only one symbol in the source word. Error propagation is therefore limited to one symbol when the p -bit prefix is received correctly, or two symbols when the p -bit prefix is received erroneously.

The sum variance of code words encoded under the rules of Knuth's algorithm was computed by Hollmann & Immink [141]. Let s_{Knuth}^2 denote the sum variance then

$$s_{\text{Knuth}}^2 = \frac{3m + 2}{16}. \quad (8.77)$$

In Example 8.4, page 226, it has been shown that the sum variance, s_0^2 , of the full set of zero-disparity codewords of length m , m even, equals, see (8.4),

$$s_0^2 = \frac{m + 1}{6}.$$

It shows that the sum variance of the set of codewords generated under the rules of Knuth's algorithm is a factor of $9/8$ larger than that of the full set of zero-disparity codewords. A slight improvement of Knuth's algorithm is possible by noting that Knuth's algorithm, as outlined above, is greedy as it takes the first opportunity for balancing the codeword. In general there is more than one position, where balancing of the words can take place. This degree of freedom can be used to pick that codeword with the minimum sum variance.

8.10 Appendix

Our objective, in this Appendix, is to derive a closed-form expression of the sum variance of dc-balanced, bi-mode channel codes. The process of encoding constituted by the alternate code principle is cyclo-stationary with period n (see Chapter 3) so that the sum variance of the sequence has to be established by averaging the sum variance over all n symbol positions within a codeword.

8.10.1 Computation of the sum variance

Let the value of the running digital sum at the k th position in a codeword be designated by z_k , and suppose also that a codeword, denoted by $\mathbf{x}^{(i)} = (x_1^{(i)}, \dots, x_n^{(i)})$ starts with an initial RDS, denoted by z_0 , one of the K positive terminal sum values (the statistics of the codewords starting at a state associated with a negative sum value can be found by symmetry). The running digital sum at the k th position in the codeword $\mathbf{x}^{(i)}$ is

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

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

$$\begin{aligned} E\{z_k^2|z_0\} &= E \left[\left(z_0 + \sum_{m=1}^k x_m^{(i)} \right)^2 \right] \\ &= E \left[z_0^2 + \sum_{m=1}^k (x_m^{(i)})^2 + 2z_0 \sum_{m=1}^k x_m^{(i)} + 2 \sum_{j_1=1}^{k-1} \sum_{j_2=j_1+1}^k x_{j_1}^{(i)} x_{j_2}^{(i)} \right], \end{aligned}$$

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

$$E\{z_k^2|z_0\} = z_0^2 + k + 2k\mu z_0 + k(k-1)r_0. \quad (8.78)$$

The sum variance of an admissible sequence that starts with initial RDS z_0 , designated by $s_K^2|z_0$, is found by averaging the running sum variance over all n symbol positions of the codeword, or

$$s_K^2|z_0 = \frac{1}{n} \sum_{k=1}^n E\{z_k^2|z_0\} = z_0^2 + \frac{n+1}{2} + \mu(n+1)z_0 + \frac{1}{3}(n^2-1)r_0.$$

The probability that a codeword starts with initial RDS $z_0 = 2i - 1$, $1 \leq i \leq K$, is π_i so that, by taking the probability into account that a codeword starts with z_0 and averaging over the $2K$ initial states, the following expression is found for the sum variance s_K^2 :

$$s_K^2 = E\{z_0^2\} + \frac{n+1}{2} + \frac{1}{3}(n^2-1)r_0 + 2(n+1)\mu \sum_{i=1}^K (2i-1)\pi_i. \quad (8.79)$$

The variance of the terminal sum values, $E\{z_0^2\}$, is

$$E\{z_0^2\} = 2 \sum_{i=1}^K (2i-1)^2 \pi_i. \quad (8.80)$$

The quantity μ can be eliminated by noting the periodicity, i.e. $E\{z_0^2\} = E\{z_n^2\}$. Evaluating (8.78) yields

$$E\{z_n^2|z_0\} = z_0^2 + n + 2n\mu z_0 + n(n-1)r_0$$

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

$$E\{z_n^2\} = E\{z_0^2\} + n + n(n-1)r_0 + 4n\mu \sum_{i=1}^K (2i-1)\pi_i,$$

so that, with $E\{z_0^2\} = E\{z_n^2\}$ we find

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

Substitution in (8.79) yields

$$s_K^2 = E\{z_0^2\} - \frac{1}{6}(n^2-1)r_0. \quad (8.81)$$

8.10.2 Computation of the correlation

We next calculate the correlation $r_0 = E\{x_{j_1}x_{j_2}\}$ of the symbols at the j_1 th and j_2 th symbol position within the same codeword. It is obvious that $E\{x_{j_1}x_{j_1}\} = 1$. If $j_1 \neq j_2$, some more work is needed. In that case

$$\begin{aligned} E\{x_{j_1}x_{j_2}\} &= Pr(x_{j_1} = x_{j_2}) - Pr(x_{j_1} \neq x_{j_2}) \\ &= 1 - 2Pr(x_{j_1} \neq x_{j_2}), \quad j_1 \neq j_2. \end{aligned} \quad (8.82)$$

Suppose a codeword to be an element of subset $S_i \subset S_+$. The probability that a symbol at position j_1 in the codeword equals '1' is

$$Pr(x_{j_1} = 1|S = S_i) = \frac{1}{n}\left(\frac{n}{2} + i\right).$$

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

$$Pr(x_{j_2} = -1 | x_{j_1} = 1, S = S_i) = \frac{\frac{n}{2} - i}{n - 1}.$$

Hence

$$Pr(x_{j_1} \neq x_{j_2} | S = S_i) = \frac{n^2 - 4i^2}{2n(n - 1)}$$

and using (8.82) yields

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

If we now take into account the probability p_i that a codeword is an element of subset $S_i \subset S_+$, then for the correlation we find

$$r_0 = E\{x_{j_1} x_{j_2}\} = \frac{-1}{n - 1} \left\{1 - \frac{4}{n} \sum_{i=1}^K i^2 p_i\right\}. \quad (8.83)$$

Combining with (8.81) yields

$$s_K^2 = 2 \sum_{i=1}^K (2i - 1)^2 \pi_i + \frac{n + 1}{6} \left\{1 - \frac{4}{n} \sum_{i=1}^K i^2 p_i\right\}. \quad (8.84)$$

Define

$$u_m = \sum_{i=1}^K i^m p_i, \quad m \in \{1, 2, 3\}. \quad (8.85)$$

The variance of the terminal sum values equals

$$E\{z_0^2\} = 2 \sum_{i=1}^K (2i - 1)^2 \pi_i,$$

which after a tedious manipulation of the various results can be written in the following elegant expression:

$$E\{z_0^2\} = \frac{4}{3} \frac{u_3}{u_1} - \frac{1}{3}. \quad (8.86)$$

After a similar manipulation we find the variance of the complete sequence

$$s_K^2 = \frac{4}{3} \frac{u_3}{u_1} + \frac{n - 1}{6} - 2 \frac{n + 1}{3n} u_2. \quad (8.87)$$

We have now completed the analysis of the classical dc-balanced codes and, with (8.63), we are in the position to easily evaluate the performance of dc-free channel codes. The analysis has not been particularly difficult because

the detailed structure of the encoder was irrelevant. The sole requirements can be summarized as follows.

The above analysis rests entirely on the assumption of the dc-balanced, bi-mode structure of the transition matrix Q . We further assumed that the expectations are invariant with respect to the location in a codeword, which is true if full codeword sets are employed.

The analysis of encoders that do not meet the above symmetry conditions is considerably more involved than that the one above presented; the detailed structure is of paramount concern. Then the performance can only be evaluated for any given code structure by resorting to numerical computational procedures, which are provided in Chapter 3.

Chapter 9

Higher-Order Spectral Zeros

9.1 Introduction

The properties of dc-balanced codes whose design rests on a digital sum constraint have been extensively dealt with in the previous chapters. The essential results can be summarized as follows. The frequency region with suppressed components, the notch width, is characterized by the cut-off frequency. As discussed, the power spectral density function of digital sum constrained codes is characterized by a parabolic shape in the low-frequency range from dc to the cut-off frequency. In fact, it is argued in Chapter 8 that the performance of simple coding schemes is not far from what is maximally feasible in terms of code redundancy and region of suppressed low-frequency components. The reader who has studied the chapters on dc-balanced codes might well assume that little remains to be added and she/he might wonder whether it is possible to escape from the strait-jacket expressed by relationship (8.34), page 213, which, in fact, shows the price-performance ratio of conventional dc-balanced codes. Is it possible to achieve, for a given redundancy, a larger rejection of the low-frequency components than is possible with conventional dc-balanced codes?

In this chapter, we present a category of dc-free codes that indeed offers a larger rejection of low-frequency components. Besides the trivial fact that they are dc-balanced, an additional property of the codes to be studied is that the second, and even higher, derivative of the code spectrum also vanishes at zero frequency (note that the odd derivatives of the spectrum at zero frequency are zero because the spectrum is an even function of the frequency). As we shall see in a moment, the imposition of this additional channel constraint results in a substantial decrease of the power at low frequencies for a fixed code redundancy as compared with the conventional designs constructed on the bounded digital sum concept.

Section 9.2 introduces determinate time-domain constraints applied, respectively, to each codeword and to the channel sequence as a whole, aim-

ing at a sequence spectrum which has both zero power and a zero second derivative at zero frequency. A sequence with these properties is said to be dc^2 -balanced. Section 9.3 gives a method to enumerate the number of dc^2 -balanced sequences. Thereafter, Section 9.4 furnishes worked examples of codes whose codewords can be freely concatenated. The power spectral density function of the dc^2 -balanced codes will be compared with those of conventional dc -balanced codes. Examples of state-dependent codes are given that operate with two modes of the source representation. Dc^2 -balanced codes are generalized in Section 9.5 to dc -balanced codes having the additional property that K low-order (even) derivatives of the code power spectrum vanish at zero frequency.

Dc -balanced codes with higher-order nulls, as discussed in this chapter, have error correcting characteristics that make them very suitable for noisy channels with a partial response transfer function. Usually in this context, dc -balanced codes are termed *matched-spectral-null codes* or *MSN codes*. A full description of the features of matched-spectral-null codes are outside the scope of this book and the interested reader is referred to the work by Monti & Pierobon, Karabed & Siegel [187], and Eleftheriou & Cideciyan [74]. The papers presented by Roth, Siegel & Vardy [292] and Tallini & Bose [313] provide a comprehensive overview of various constructions and bounds.

9.2 Preliminaries

Let $\{x_i\}$ denote a stationary channel sequence of bipolar symbols, having mean zero, with variables $x_1, \dots, x_i \in \{-1, 1\}$. The power spectral density function of the sequence is given by (see Chapter 3)

$$H(\omega) = R(0) + 2 \sum_{i=1}^{\infty} R(i) \cos i\omega, \quad (9.1)$$

where $R(i) = E\{x_j x_{j+i}\}$, $i = 0, \mp 1, \mp 2, \dots$ is the auto-correlation function of the sequence. The spectrum $H(\omega)$ is an even function of ω , so that in the neighborhood of $\omega = 0$ it can be approximated with the Taylor expansion

$$H(\omega) = H(0) + \frac{1}{2}H^{(2)}(0)\omega^2 + \frac{1}{24}H^{(4)}(0)\omega^4 \dots \quad (9.2)$$

A sequence is said to have a K th order spectral-null if its spectrum $H(\omega)$ has a null at zero frequency, i.e., $H(0) = 0$, and if its derivatives at zero frequency $H^{(2i)} = 0$, $i = 1, \dots, K$.

In this chapter, we will discuss various constructions of codes that generate sequences with a K th order spectral null. We start with the simplest case, namely where $K = 1$. Thereafter, we will outline the general case.

The running digital sum (RDS), denoted by z_i , is defined by (see Section 8.2)

$$z_i = \sum_{j=1}^i x_j. \quad (9.3)$$

As shown in Section 8.2, the power spectral density function vanishes at $\omega = 0$ if the sequence $(z_i)_{i=1}^{\infty}$ is bounded, that is, if for all i $|\{z_i\}| < M_z$, where M_z is an arbitrary positive constant. We now define the running digital sum sum (RDSS) y_i by

$$y_i = \sum_{j=1}^i z_j. \quad (9.4)$$

Provided both the RDS and RDSS are bounded, that is, for all i

$$|\{z_i\}| < M_z \text{ and } |\{y_i\}| < M_y,$$

it can be proved in the same way that then both $H(0) = 0$ and $H^{(2)}(0) = 0$. In the ensuing sections a technique is presented for devising codes that possess the virtues of bounded RDS and RDSS. These codes will be termed dc^2 -balanced codes. The type of code presented here is of fixed length and it is state-independently decodable. The codewords are selected in such a way that after concatenation, under certain encoder rules, the channel sequence will assume a bounded RDS and RDSS. The first problem we will tackle, to be considered in the next Section, is the counting of the number of codewords of length n that conform to prescribed y and z constraints.

9.3 Enumeration of sequences

Up till now, the word unbalance referred to a property of a sequence of arbitrary length. Now we shall use this concept for codewords of predefined finite length n . First, simple recursion relations are derived to enumerate the number of codewords of length n having a constraint on both the z and y disparity. These sequences are termed (z, y) sequences. Later we will demonstrate how these codewords can be concatenated in such a way that the concatenated code stream assumes a limited number of RDS and RDSS values. Specifically, we will enumerate the number of sequences $\mathbf{x} = (x_1, \dots, x_n)$, $x_i \in \{-1, 1\}$, that meet the conditions

$$\begin{aligned} \sum_{i=1}^n x_i &= d_z \\ \sum_{j=1}^n \sum_{i=1}^j x_i &= d_y, \end{aligned} \quad (9.5)$$

where the given constants d_z and d_y are termed the z and y disparities of \mathbf{x} . By changing the order of summation we obtain

$$d_y = \sum_{i=1}^n (n+1-i)x_i = (n+1)d_z - \sum_{i=1}^n ix_i. \quad (9.6)$$

If $d_z = 0$ and $d_y = 0$, we simply find that the codeword must satisfy the conditions

$$\sum_{i=1}^n x_i = \sum_{i=1}^n ix_i = 0. \quad (9.7)$$

We shall now show that n must be a multiple of four in order to satisfy the above conditions $d_z = 0$ and $d_y = 0$. In view of $x_i \in \{-1, 1\}$, we may rewrite (9.6) and obtain

$$d_y = - \sum_{i=1}^{(n+d_z)/2} p_i + \sum_{i=1}^{(n-d_z)/2} n_i + (n+1)d_z, \quad (9.8)$$

where $p_i \in \{1, \dots, n\}$ and $n_i \in \{1, \dots, n\}$ correspond to the positions of the +1s and -1s, respectively. Obviously,

$$\sum_{i=1}^{(n+d_z)/2} p_i + \sum_{i=1}^{(n-d_z)/2} n_i = \sum_{i=1}^n i = \frac{1}{2}n(n+1),$$

so that, using (9.8), we find

$$4 \sum_{i=1}^{(n+d_z)/2} p_i = n(n+1) + 2(n+1)d_z - 2d_y. \quad (9.9)$$

Thus if $d_z = 0$ and $d_y = 0$, it follows in a straightforward manner from (9.5) and (9.9) that the codeword length n must be a multiple of four.

The number of codewords that meet given d_y and d_z conditions can be computed using the theory of partitions (see Riordan [290]). From the above, it follows that the number of codewords which comply with the given restrictions equals the number of subsets $\{p_1, \dots, p_{(n+d_z)/2}\}$ of size $(n+d_z)/2$ of the set $\{1, 2, \dots, n\}$ satisfying (9.9). The number of codewords, $A_n(d_z, d_y)$, is given by the coefficient c_n of $u^{i_0}t^{j_0}$ of the polynomial $g_n(u, t)$ defined by

$$g_n(u, t) = (1+ut)(1+u^2t) \dots (1+u^nt) = \sum_{i,j} c_n(i, j)u^i t^j, \quad (9.10)$$

where

$$i_0 = \{n(n+1) + 2(n+1)d_z - 2d_y\}/4,$$

and

$$j_0 = (n+d_z)/2.$$

Both i and j are assumed to be integers, otherwise the number of codewords is zero.

A useful relation can be derived with the following observation. The sequence $\mathbf{x} = (x_1, \dots, x_n)$ satisfies the d_z and d_y constraint if the sequence \mathbf{w} obtained by reversing the order of symbols in \mathbf{x} , i.e. $\mathbf{w} = (x_n, \dots, x_1)$, satisfies the constraints

$$\sum_{i=1}^n w_i = d_z, \quad \sum_{j=1}^n \sum_{i=1}^j w_i = -d_y + (n+1)d_z.$$

By definition, the number of sequences \mathbf{w} that meet the two conditions equals $A_n(d_z, (n+1)d_z - d_y)$. Since the number of sequences \mathbf{w} equals the number of sequences \mathbf{x} with the given constraints, we obtain

$$A_n(d_z, d_y) = A_n(d_z, (n+1)d_z - d_y). \quad (9.11)$$

Since $g_n(u, t) = g_{n-1}(u, t)(1 + u^n t)$ the coefficients $c_n(i, j)$ can be found recursively:

$$c_m(i, j) = c_{m-1}(i, j) + c_{m-1}(i - m, j - 1), \quad (9.12)$$

with initial conditions $c_0(0, 0) = 1$, otherwise $c_0(i, j) = 0$. Alternative enumeration schemes have been developed by Xin & Fair [352].

9.4 Coding with zero-disparity codewords

A natural way to construct a code that generates long sequences satisfying the z and y constraints is to employ a set of codewords of fixed length n that can be concatenated without a coding rule, that is, to employ a memoryless encoder. In this elementary case, the codewords start and terminate with both zero RDS and RDSS and consequently the codewords should have both zero z and y disparities. In the sequel, the shorthand notation zero-disparity codeword is used if both the z and y disparities of that codeword are zero. From (9.7), we find in this particular case that the codeword moments, designated by u_k , meet the following conditions:

$$u_k = \sum_{i=1}^n i^k x_i = 0, \quad k \in \{0, 1\}. \quad (9.13)$$

Invoking recursion relations (9.12) we calculated the number $M = A_n(0, 0) = c_n(n(n+1)/4, n/2)$, $n = 4, 8, \dots, 36$ of zero-disparity codewords (we have already pointed out that the set of zero-disparity codewords is empty if n is not a multiple of 4). Results of computations are listed in Table 9.1.

Table 9.1: Number of zero-disparity codewords and rate versus codeword length n .

n	M	R
4	2	0.250
8	8	0.375
12	58	0.488
16	526	0.565
20	5448	0.621
24	61108	0.662
28	723354	0.695
32	8908546	0.722
36	113093022	0.743

Table 9.1 also presents the code rate R , which is defined by

$$R = \frac{1}{n} \log_2 M. \quad (9.14)$$

It can be noticed that the code rate, even for comparatively large codeword length n , is quite poor. It was shown by Saxena & Robinson [294] and Tallini & Bose [313] that for large n the number of zero-disparity codewords can be approximated by

$$M \approx \frac{4\sqrt{3}}{\pi} \cdot \frac{2^n}{n^2}.$$

This asymptotic formula implies the following approximation for the rate R :

$$R \approx 1 - \frac{1}{n} \{2 \log_2 n - 1.141\}, \quad n \gg 1. \quad (9.15)$$

Numerical results are given in Table 9.2.

Table 9.2: Rate (approximated) of zero-disparity codewords versus codeword length n .

n	R	n	R
8	0.3925	256	0.9420
16	0.5713	512	0.9671
32	0.7231	1024	0.9816
64	0.8303	2048	0.9898
128	0.8995	4096	0.9944

9.4.1 Spectra of zero-disparity codewords

Let us now examine the power spectral density function of codes based on fixed-length zero-disparity codewords. Let $\mathbf{x}^{(k)} = (x_1^{(k)}, \dots, x_n^{(k)})$, denote the k th element of the set S of zero-disparity codewords in $\{-1, 1\}$. If, as usual, it is assumed that the codewords are equiprobable and independent, then the auto-correlation function of the concatenated channel stream, when transmitted serially, is given by (see Chapter 3)

$$\begin{aligned} R(k) &= \frac{1}{n} \sum_{i=1}^{n-k} E\{x_i x_{i+k}\}, \quad 0 \leq k \leq n-1, \\ R(0) &= 1, \\ R(k) &= 0, \quad k \geq n, \end{aligned}$$

where

$$E\{x_i x_j\} = \frac{1}{M} \sum_{k=0}^{M-1} x_i^{(k)} x_j^{(k)}, \quad 1 \leq i, j \leq n. \quad (9.16)$$

Example 9.1 The simplest example is the code constituted by codewords of length $n = 4$. We can verify that the two codewords are '+ - + -' and '+ - - +' (the characters '+' and '-' are used to represent a bipolar symbol with value +1 and -1, respectively). The code rate is $R = 1/4$. The auto-correlation function of this channel code is $R(0) = 1$, $R(1) = -1/4$, $R(2) = -1/2$, $R(3) = 1/4$, and $R(i) = 0$, $i \geq 4$. The power spectral density function $H(\omega)$ of this channel code is readily established with (9.1):

$$\begin{aligned} H(\omega) &= 1 + 2\left\{-\frac{1}{4} \cos \omega - \frac{1}{2} \cos 2\omega + \frac{1}{4} \cos 3\omega\right\} \\ &= 4 \sin^2 \frac{\omega}{2} \sin^2 \omega. \end{aligned}$$

Without much difficulty we may verify that indeed $H(0) = H^{(2)}(0) = 0$.

Although the execution of expression (9.16) is straightforward, it has the disadvantage that, due to the exponential growth of the number of codewords, the arithmetical effort becomes prohibitive for large codeword lengths. This problem is being tackled by exploiting the preceding enumeration method, which, by contrast, has the virtue that the auto-correlation function can be computed with a polynomially growing computational workload. To that end, let i_0 and i_1 , $i_0 \neq i_1$ be two symbol positions in a codeword. From (9.16), we derive for fixed i_0 and i_1 , using $x_i \in \{-1, 1\}$:

$$E\{x_{i_0} x_{i_1}\} = \frac{1}{M} \{N(x_{i_0} = x_{i_1}) - N(x_{i_0} \neq x_{i_1})\}, \quad i_0 \neq i_1,$$

where $N(\cdot)$ is the number of codewords satisfying (\cdot) . For reasons of symmetry:

$$\begin{aligned} E\{x_{i_0}x_{i_1}\} &= \frac{2}{M}N(x_{i_0} = x_{i_1}) - 1 \\ &= \frac{4}{M}N(x_{i_0} = x_{i_1} = 1) - 1. \end{aligned} \quad (9.17)$$

Let $x_{i_0} = x_{i_1} = 1$, then from (9.13)

$$\sum_{i=1, i \neq i_0, i_1}^n x_i = -2$$

and

$$\sum_{i=1, i \neq i_0, i_1}^n ix_i = -(i_0 + i_1). \quad (9.18)$$

The number of codewords in S that meet the above conditions can again be found with the enumeration method described in Section 9.3. The number of sequences $N(x_{i_0} = x_{i_1} = 1)$ in the codeword set S that satisfy (9.18) is given by the coefficient of $u^i v^j$, where $i = n(n+1)/4 - i_0 - i_1$, $j = (n-4)/2$, of the polynomial

$$h_n(u, t) = \frac{g_n(u, t)}{(1 + u^{i_0}t)(1 + u^{i_1}t)}. \quad (9.19)$$

With a small modification of the recursion relations (9.12), we obtain $N(x_{i_0} = x_{i_1} = 1)$. Using (9.1), (9.16), (9.17), and relation (9.19) we calculated the power spectral density function of zero-disparity codeword based codes. The outcomes of the computations are plotted in Figure 9.1.

As with standard dc-balanced codes, we can define the cut-off frequency of dc²-balanced codes. A computation of the cut-off frequency of dc-balanced and dc²-balanced zero-disparity codes for a fixed rate, leads to the conclusion that the cut-off frequency of conventional dc-balanced codes exceeds that of dc²-balanced codes by a factor of 2.5. It should be borne in mind, however, that below the cut-off frequency, the spectrum of dc²-balanced codes decreases faster with decreasing frequency than those of dc-balanced codes. We noticed when comparing the spectra of dc²- and dc-balanced codes that for fixed redundancy a cross-over is found at approximately -20 dB. Accordingly, we infer that dc²-balanced codes are favorable if a rejection of low-frequency components better than -20 dB is required. A few remarks regarding the construction of dc²-balanced zero-disparity codes are in order. As we may observe in Table 9.1, the rate of dc²-balanced block codes is quite poor for relatively short codewords. In order to increase the rate, we must seek resort to the usage of (very) long codewords, where direct look-up is impractical. Using enumeration techniques, as discussed in Chapter 6, it is straightforward to encode an arbitrary source word into a dc²-balanced zero-disparity codeword.

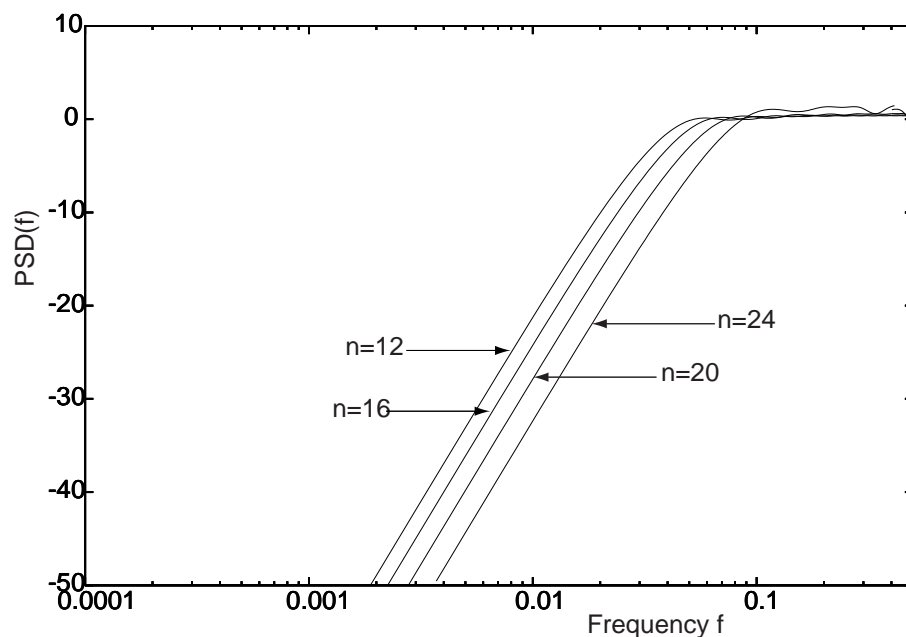


Figure 9.1: Power spectral density function, $H(\omega)$, of zero-disparity dc^2 -balanced codes with the codeword length $n = 12, 16,$ and 20 , as a parameter.

Unfortunately, the number of states (weights) increases strongly, (n^3), with increasing codeword length n , and enumeration is therefore not a very attractive option. Siegel & Vardy [301] disclosed an "encoding method and apparatus" for constructing an asymptotically optimal coding scheme for second-order dc-free constrained channels. Roth *et al.* [292] describe a construction, where by inverting a very limited number, $\approx 3 \log_2 n$, of source symbols, a dc^2 -balanced zero-disparity codeword can be obtained. The modifications of the source word are sent encoded as a zero-disparity prefix word to the receiver, which can then undo the modifications made.

9.5 State-dependent encoding

In the case of state-independent (memoryless) encoding, the sequence is encoded without information of the past. This format implies a unique one-to-one mapping of the source words and their channel representations. A larger rate with given codeword length is feasible when codewords are allowed to start (and end) in a state that is a member of a set of predefined states, commonly referred to as principal states. The existence of such a set of principal states can be ascertained with a routine developed by Franaszek (see also Chapter 5).

During experiments with this procedure, we found that the following

choice of codeword subsets is quite satisfactory. It is assumed that the encoder operates on the basic principle of the bi-mode alternate coding technique as described by Cattermole (see Chapter 8). All codewords are chosen in such a way that $d_z = 0$. Codewords with positive and negative d_y are used in two modes, and codewords with zero d_y may serve in both modes. The polarity of a codeword is to be chosen by the encoder in such a way that at the end of each new codeword the RDSS is as close to zero as possible. The number of RDSS values at the end of a codeword is twice the number of codeword subsets with different positive d_y values. The RDSS at the end of the codewords will assume the values $\mp 1, \mp 3, \dots, \mp(r-1)$ by initializing the RDSS counter with the disparity 1. The recursion relations can be used to calculate the rate of the bi-mode code as a function of the number of subsets and the length of the codeword. Table 9.3 lists the outcomes when r encoder states are used. Note the improvement in code rate with respect to those of the memoryless block codes listed in Table 9.1. The performance of two rate $R = 1/2$ codes will be assessed in the following examples.

Table 9.3: Code rate R of dc²-balanced codes with codeword length n and r encoder states.

$r =$	2	4	6	8	10
$n = 4$.396	.500	–	–	–
8	.488	.557	.594	.625	.641
12	.568	.616	.648	.672	.689
16	.627	.663	.688	.707	.722
20	.670	.700	.720	.736	.748

Example 9.2 An attractive and conceptually simple code shown in Table 9.3 has the parameters $n = 4$, $r = 4$, and $R = 1/2$. Table 9.4 shows the codebook of this simple code. The codewords contain equal numbers of +1s and -1s which facilitates the implementation of the encoder as it suffices to monitor the RDSS.

Table 9.4: Codebook of four-state $R = 1/2$ code.

i	$h, g(\sigma_1, \beta_i)$	$h, g(\sigma_2, \beta_i)$	$h, g(\sigma_3, \beta_i)$	$h, g(\sigma_4, \beta_i)$
0	1001, σ_1	1001, σ_2	1001, σ_3	1001, σ_4
1	0110, σ_1	0110, σ_2	0110, σ_3	0110, σ_4
2	1010, σ_2	1010, σ_3	0101, σ_2	0101, σ_3
3	1100, σ_3	1100, σ_4	0011, σ_1	0011, σ_2

The codewords should be chosen from the table in such a way that after transmission of the new codeword the RDSS is as close to zero as possible. The RDSS at the end of each transmitted codeword assumes four values, namely ∓ 1 and ∓ 3 .

Example 9.3 A rate $R = 1/2$, three-state encoder which, by virtue of the reduced RDSS variation, slightly (approximately 2 dB) improves upon the low-frequency content of the four-state code studied in Example 9.2 was published by Monti *et al.* [252].

Table 9.5: Codebook of three-state $R = 1/2$ code.

i	$h, g(\sigma_1, \beta_i)$	$h, g(\sigma_2, \beta_i)$	$h, g(\sigma_3, \beta_i)$
0	0110, σ_1	0110, σ_2	0110, σ_3
1	1001, σ_1	1001, σ_2	1001, σ_3
2	1010, σ_2	1010, σ_3	0011, σ_1
3	1100, σ_3	0101, σ_1	0101, σ_2

The complete code specification, in terms of the output and state-transition functions, is given in Table 9.5. The three encoder states are denoted by σ_i , $i = 1, 2, 3$. The codewords have been allotted in such a way that decoding turns out to be state independent, while the assignment with the source words is arbitrary. The reader may easily verify that in the above scheme the same codewords are used as in the code described by Example 9.2. The more judicious choice of the code structure, however, accounts for the improved performance at the low-frequency end.

9.6 Higher-order dc-constrained codes

The concept of dc²-constrained sequences established in the preceding section can be generalized to include higher-order nulls at the zero frequency [160]. Define the following $K + 1$ moments of a codeword:

$$u_k = \sum_{i=1}^n i^k x_i, \quad k \in \{0, 1, \dots, K\}. \quad (9.20)$$

A codeword is defined as being of K th order zero-disparity if

$$u_k = 0, \quad k \in \{0, 1, \dots, K\}. \quad (9.21)$$

The set of all K th order zero-disparity codewords of length n is called the K th order disparity code of length n . It can easily be verified that the first $2K + 1$ derivatives of the power spectral density function of a K th

order zero-disparity code vanish at $\omega = 0$. There is an interesting analogy between this relationship and its counterpart in signal theory. It is well known that if the first $K + 1$ moments m_0, m_1, \dots, m_K of a function are zero, then also the first $K + 1$ derivatives of its Fourier transform vanish at zero frequency [270]. In that case, it can be verified in a straightforward manner that the first $2K + 1$ derivatives of its power spectrum also vanish at $\omega = 0$.

In the simple case $K = 0$, we find that the length n of the codewords is even and that the codewords obviously have an equal number of \mp 1s. The number of such codewords M is given by the binomial coefficient

$$M = \binom{n}{\frac{n}{2}}. \quad (9.22)$$

The properties of this category of dc-balanced block code are treated in Chapter 8. If $K = 1$, it was shown in the previous section that each codeword corresponds to a partition of the set $\{1, \dots, n\}$ into two disjoint subsets of size $n/2$, such that the sum of the elements in both sets equals $n(n+1)/4$.

Let \mathbf{x} be a codeword and let I be the set of indices l for which $x_l = 1$ and let J be the set of indices l for which $x_l = -1$. Then from the fact that \mathbf{x} is a first-order zero-disparity codeword, it follows that

$$\sum_{i \in I} i = \sum_{j \in J} j = \frac{1}{2} \sum_{l=1}^n l = \frac{n(n+1)}{4}. \quad (9.23)$$

Hence I and J divide the set $\{1, \dots, n\}$ into two disjoint subsets with the required properties. These two sets correspond to the position of the "1"s and "-1"s, respectively, in the codewords. Conversely, each such partition corresponds to a codeword. Hence, the number of codewords equals the number of ways to partition the set $\{1, 2, \dots, n\}$ into two disjoint subsets of size $n/2$, such that the sum of the elements in both sets equals $n(n+1)/4$. Using generating functions, (see Section 5.9) the number of codewords with $u_0 = u_1 = 0$ can be expressed as the coefficient $c_n(i, j)$ of $u^i t^j$, $i = n(n+1)/4$, $j = n/2$ in the polynomial $g_n(u, t)$ defined by

$$g_n(u, t) = (1 + ut)(1 + u^2t) \dots (1 + u^nt). \quad (9.24)$$

It is assumed that n is a multiple of four.

In general, it can be shown [160] that the number of codewords of a K th order zero-disparity code is the coefficient of

$$v_0^{i_0} v_1^{i_1} \dots v_K^{i_K}$$

of the polynomial $G_n(v_0, v_1, \dots, v_K)$ defined by

$$G_n(v_0, v_1, \dots, v_K) = \prod_{i=1}^n (1 + v_0 v_1^i \dots v_K^{iK}), \quad (9.25)$$

where

$$i_m = \frac{1}{2} \sum_{i=1}^n i^m.$$

Note that for $K = 1$, (9.25) reduces to (9.24). Table 9.6 lists the number of K th order zero-disparity codewords versus codeword length n and the order K for $K = 1, 2, 3$, and 4.

Table 9.6: Number of K th order zero-disparity codewords versus code-word length n and K .

n	$K = 1$	$K = 2$	$K = 3$	$K = 4$
4	2	0	0	0
8	8	2	0	0
12	58	2	0	0
16	526	14	2	0
20	5448	48	0	0
24	61108	592	16	0
28	723354	2886	0	0
32	8908546	34888	78	2

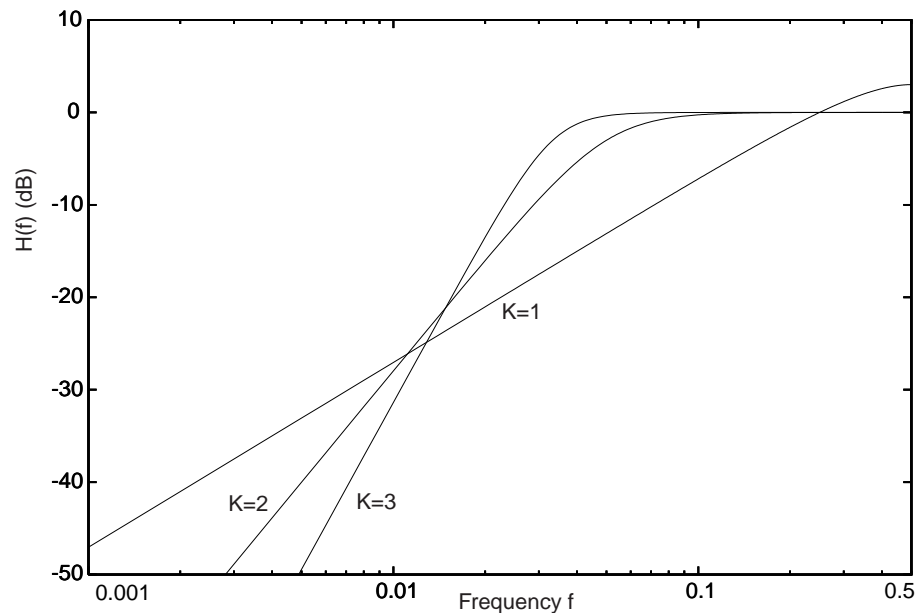


Figure 9.2: Power density functions of memoryless codes with $K = 0, 1$, and 2 with rate $1/2$. For increasing values of the order K , we obtain a more severe suppression of the power at very low frequencies.

The spectra of codes based on K th order zero-disparity codewords for $K = 0, 1,$ and 2 whose codeword lengths are chosen, see Table 9.6, in such a way that the rate R of these codes is approximately $1/2$, are illustrated in Figure 9.2. It can be seen in the diagram that for increasing values of the order K , we obtain a more severe attenuation of the power at very low frequencies. Skachek *et al.* described an efficient encoding algorithm for third-order spectral-null codes.

Asymptotic behavior in n of the number of K th order zero-disparity codewords was determined by Freiman & Litsyn [102]. They showed that, for sufficiently large n , the rate R of the K th order zero-disparity codeword sets can be approximated by

$$R \approx 1 - \frac{(K+1)^2}{2n} \log_2 n, \quad n \gg 1. \quad (9.26)$$

Note that the above relationship coincides with those of $K = 0$, regular zero-disparity codewords, see (8.38) at page 214, and $K = 1$, DC²-balanced codewords, see (9.15), page 248.

Chapter 10

Guided Scrambling

10.1 Introduction

In both coding practice and the literature, code implementations have been concentrated on codes whose codeword length is relatively small. Examples are the rate $1/2$, rate $2/3$ RLL codes, byte-oriented dc-free codes of rate $8/10$ or $8/9$ (see Chapter 8), and byte-oriented RLL codes such as EFM and EFMPlus (see Chapter 11). The hard disk drive (HDD) industry has been moving towards detection schemes that can function well at very high code rate such as $16/17$, $24/25$, and $32/33$. The construction of high-rate codes is far from obvious, as table look-up for encoding and decoding is an engineering impracticality. In much of the previous work, where look-up tables are used, the difficulty of looking-up has been attacked by a "divide-and-conquer" approach. Alternatively, the high-rate $(0, k)$ codes discussed in Section 6.3.4, page 150, which use enumerative coding schemes, offer a viable solution when the codeword length is very long.

Coding methods that do not use table look-up or enumeration are of specific practical interest. As examples we mention two methods, described in Chapter 8, which are good candidates for high-rate dc-free codes, namely the polarity-bit code by Bowers and the dc-balancing method by Knuth. Both methods exploit the idea that the correspondence between source words and the codewords should be as simple as possible. Usually this means that the encoded bits are as much as possible equal to the source bits, or inverted versions of it. Clearly, equality or inversion are the simplest operations possible. As discussed in Chapter 8, a drawback of said methods is that the performance, in terms of suppression of low-frequency components, is far from what could be obtained according to the tenets of information theory. Up till now, attempts to improve the performance of such codes failed as "good" codes require large look-up tables.

The publications by Fair *et al.* [76, 77, 78] and Immink & Patrovics [166] on "guided scrambling" stimulated new research in this area. Guided scram-

bling is a member of a larger class of related coding schemes called *multi-mode* code. In multi-mode codes, each source word can be represented by a member of a selection set consisting of L codewords. The encoder evaluates the "quality" of each codeword in the selection set, and transmits that codeword that "best" matches the quality criterion, metric, at hand. At first sight, this does not look as a very novel approach, as it has been 'business-as-usual' since the advent of constrained codes. The difference, however, with the classical approach as discussed in the previous chapters, is that the members of the selection sets are not judiciously selected and stored in memory. They are, in contrast, randomly picked. The basic idea is that, provided the selection set is sufficiently large, we will find, with high probability, an adequate codeword fulfilling the constraints at hand. The advantages of this approach are evident: at the encoder site we only need a) a simple mechanism for translating source words into a selection set of "random" codewords and b) a mechanism for evaluating the "quality" of the candidate words. The stumbling block of conventional large codes, the huge look-up table, is avoided, and replaced by a simple randomization algorithm. At the receiver's site we only need a de-randomization procedure, which, as we will see in the next section, can often be embodied by a simple sliding-block decoder so that error propagation is limited. Clearly, the search and computational load for evaluating the metric at the encoder's site puts a practical limit to the size of the selection set. It should be noted that the decoder does not search or evaluate, so that this scheme is of particular interest to broadcasting or optical recording on mass-replicated discs. Embellishments for reducing the computational load, such as a non-exhaustive search of the selection set, can easily be added to the generic scheme. In the context of dc-free codes, Copeland & Tezcan [64] showed that a significant saving in computational load can be gained by employing a Fast (Walsh-)Hadamard Transform. The Hadamard transform generates the possible disparity values of all n candidate codewords in the selection set in $\log n$ instead of n steps (see Section 10.4). The two key ingredients of the encoder mentioned above need to be chosen judiciously:

1. the random mapping between the source words and their corresponding selection sets, and
2. the metric used to select the "best" word from the selection set.

The spectral (or other) performance of the code greatly depends on both issues. We will in this chapter discuss results of the application of guided scrambling to both RLL and dc-free codes.

Provided the selection size, L , is large enough and that the selection set contains sufficiently different codewords, multi-mode codes can be used to almost satisfy any channel constraint. It is, however, not possible to fully

guarantee the specified constraints. There is always a (hopefully small) probability that the specified constraints will be violated. Codes that do not strictly guarantee the fulfillment of specified constraints will be called *weakly constrained codes*. Weakly constrained codes produce sequences that violate the specified constraints with probability p . The fact that the given rules are not observed is not thought to be very devastating as also the recording channel is not free of errors. Clearly, if the norm gets looser, the channel capacity will increase, and this will make it, hopefully, easier to construct codes.

The outline of this chapter is as follows. We start, in Section 10.2, with a general outline of guided scrambling. Thereafter, In Section 10.3, we will focus on the spectral characteristics of dc-balanced codes that are generated under the rules of the guided scrambling algorithm. In Section 10.4, we will present a possible implementation for efficiently evaluating a selection set. Thereafter, in Section 10.5, we compare the performances of guided scrambling and the polarity bit scheme using look-ahead.

In Section 10.6, we study the performance of weakly $(0, k)$ constrained codes. We will compute the probability, P , that the encoder fails to transmit a codeword that complies with the specified k constraint as a function of various coding parameters such as redundancy and size of the selection set. It will be shown that, under mild conditions, we have that $\ln(P)$ is proportional to $-2^{(C-R)n}$, so that, as long as $R < C$ we can, in principle, decrease the encoder failure rate by choosing the codeword length (and the selection set) sufficiently large.

10.2 Guided scrambling basics

A basic element of multi-mode codes is the definition of a one-to- L invertible mapping between the source word \mathbf{x} and its selection set. The invertible mapping should be designed in such a way that the selection sets contain sufficiently 'random' codewords, and that error propagation at the decoding site is limited.

Examples of good mappings that have the above desirable attributes are the guided scrambling algorithm presented by Fair *et al.* [76], the dc-free coset codes of Deng & Herro [70], the multiple scramblers of Kanota & Nagai [185], and the scrambling using a linear error correcting code, such as the Reed-Solomon code, by Denissen & Tolhuizen [71], the Hadamard Transform by Copeland & Tezcan [64], and Kunisa *et al.* [212-213]. In our context, a mapping is considered to be "good" if, indeed, the selection sets contain sufficiently distinct codewords.

10.2.1 Guided scrambling

Scrambling techniques have traditionally been employed in digital transmission and recording systems to 'randomize' the source words. Scrambling is a very simple operation: it is usually accomplished by adding (modulo-2) a known pseudo-random sequence to the source sequence. The pseudo-random sequence is known to receiver and sender. The pseudo-random sequence can be stored in memory, or it can, as is usually done, generated on-the-fly with a feedback shift register. There are many good techniques for translating a user word into a selection set of 'random' codewords. Kanota & Nagai [185] use a plurality of scramblers each of which scrambles the input data. The scrambler that generates the "best" output is chosen and identified in the message sent. A disadvantage of this method is error propagation since the entire message will be received in error if the data identifying the scrambler used is erroneously received. The next method, using a self-synchronizing scrambler, solves this problem.

With judiciously chosen feedback taps, the feedback shift register will generate a maximal-length sequence called an *m-sequence*. The source stream is simply recovered at the receiver's site (this operation is called *de-scrambling*) by adding the same pseudo-random sequence to the received message. A scrambler using a feedback shift register is often termed self-synchronizing. Basic theory of *self-synchronizing* scramblers can be found in [218]. The term 'self synchronizing' stems from the fact that no frame synchronization or resetting of the scrambler and descrambler (at the receiver's site) is required for proper operation (see later for a description of the descrambler). Note that frame synchronization is normally required when scrambling is done by adding a known, random, sequence.

A self-synchronizing scrambler comprises a (binary) shift register of length s , whose input consists of the addition of s_w , $s_w \leq s$, delayed versions of the output symbol. A diagram of such a self-synchronizing scrambler is shown in Figure 10.1. All additions are assumed to be modulo-2 additions, and the addition operation is denoted by the symbol \oplus . Commonly, the *scrambler polynomial* is used to denote which delayed versions of the output are fed back to the shift register input. Let the scrambler polynomial be given by

$$x^s + \sum_{k=1}^s a_k x^{s-k},$$

where $a_k \in \{0, 1\}$, $k = 1, \dots, s$, denote the scrambler coefficients. If $a_k = 1$, the k cycles delayed version of the output is fed back to the input. Let A be the set of integers k for which $a_k = 1$. The cardinality of A , i.e., the number of feedback taps, is denoted by s_w , and is called the *weight* of the scrambler polynomial. The sequence that is generated by the scrambler is periodic. For a given length, s , of the shift register, the period can be

at most $2^s - 1$ [112]. A sequence with this maximum period is called a *maximum-length sequence* or *pseudo random sequence*.

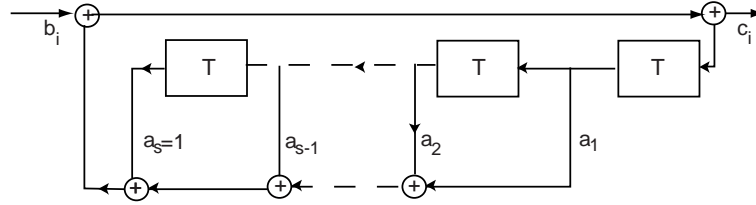


Figure 10.1: Example of self-synchronizing scrambler. At each instant i , a (binary) symbol b_i is forwarded to the scrambler, and an output, scrambled, symbol c_i is delivered.

In the first step, called *augmenting*, the source word $\mathbf{x} = (x_1, \dots, x_m)$ is preceded by all the possible binary sequences of length r to produce the intermediate set $\mathcal{B}_x = \{\mathbf{b}_1, \dots, \mathbf{b}_L\}$. Hence:

$$\begin{aligned} \mathbf{b}_1 &= (0, 0, \dots, 0, 0, x_1, \dots, x_m), \\ \mathbf{b}_2 &= (0, 0, \dots, 0, 1, x_1, \dots, x_m), \\ &\dots \\ \mathbf{b}_L &= (1, 1, \dots, 1, 1, x_1, \dots, x_m). \end{aligned}$$

A diagram of the above augmenting process is shown in Figure 10.2. The selection set $\mathcal{C}_x = \{\mathbf{c}_1, \dots, \mathbf{c}_L\}$ is obtained by scrambling all vectors in the intermediate set \mathcal{B}_x . The scrambler translates each vector $\mathbf{b} = (b_1, \dots, b_n) \in \mathcal{B}_x$ into $\mathbf{c} = (c_1, \dots, c_n) = f(\mathbf{b}) \in \mathcal{C}_x$ using the recursion

$$c_i = b_i \oplus \sum_{k \in A} c_{i-k}. \tag{10.1}$$

The “best” codeword in \mathcal{C}_x is selected for transmission. At the receiver’s site, the inverse operation $\mathbf{b} = f^{-1}(\mathbf{c})$ is

$$b_i = c_i \oplus \sum_{k \in A} c_{i-k}.$$

The above operation can be seen in Figure 10.3.

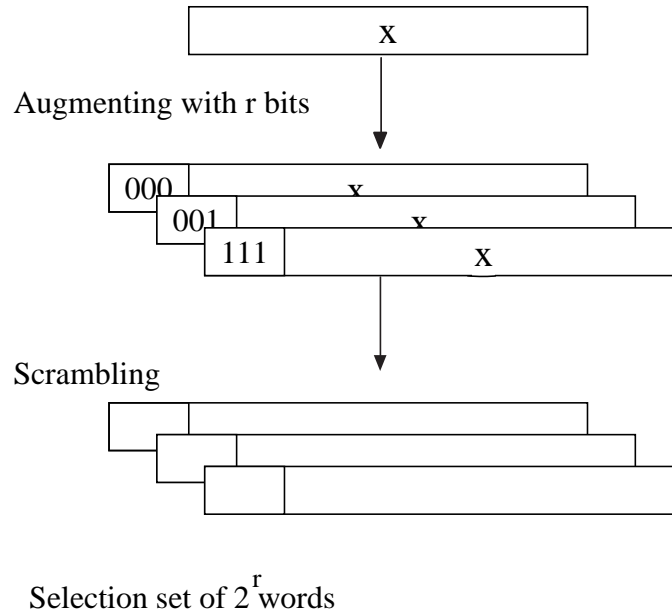


Figure 10.2: Process of augmenting using guided scrambling. A set of size 2^r $(m + r)$ -bit words is constructed by augmenting the m -bit input word with all possible r -bit words. Each $(m + r)$ -bit word is scrambled, thus generating a selection set of 2^r 'randomized' words.

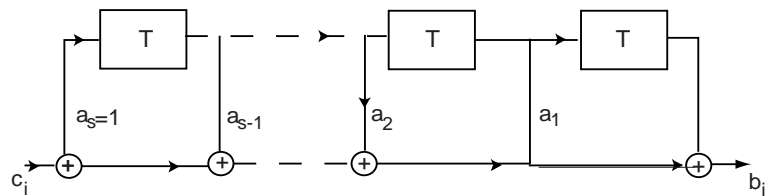


Figure 10.3: Descrambler. The output symbol b_i is simply found by the addition of delayed versions of the input symbol c_i .

Clearly the above operation does not require any synchronization of the scrambler or descrambler. The source word is simply found by deleting the first r bits. Note that the descrambling operation is essentially a sliding-block decoder of length s , where $s_w + 1$ modulo-2 additions are made for retrieving a source symbol. As a consequence, if a single channel error has been made in the received word, then, after descrambling, $s_w + 1$ errors will be propagated in the descrambled word. The weight s_w of the scrambler polynomial (the number of taps) is therefore a key parameter for determining the error propagation. Good scrambler polynomials with a minimum weight have been tabulated, see, for example, Peterson & Weldon [282].

10.3 Analysis of multi-mode dc-free codes

In this section, we will present results of the application of guided scrambling to generate dc-balanced sequences.

As discussed above, in the guided scrambling algorithm, translation of source words into 2^r random-like channel representations is done in a fairly simple way. This basic algorithm is, however, prone to worst case situations since there is a probability that consecutive source words have representation sets whose members all have the same polarity of the disparity. In this vexatious situation, the *running digital sum* RDS cannot be controlled, and long-term low-frequency components can build up. This flaw can be solved by a construction where each selection set consists of pairs of words of opposite disparity. As a result, there is always a codeword in the selection set that can control the RDS.

A simple method embodying this idea combines the features of guided scrambling and the polarity bit code. The improved algorithm, using $r \geq 2$ redundant bits, is executed in six steps. In Steps 1, 2, and 5 the original guided scrambling principle is executed while Steps 3 and 4 embody the polarity bit code.

1. The source word \mathbf{x} is preceded by all the possible binary sequences of length $(r - 1)$ to produce the $L' = 2^{r-1}$ elements of the set $\mathcal{B}_x = \{\mathbf{b}_1, \dots, \mathbf{b}_{L'}\}$. Hence:

$$\begin{aligned} \mathbf{b}_1 &= (0, 0, \dots, 0, 0, x_1, \dots, x_m), \\ \mathbf{b}_2 &= (0, 0, \dots, 0, 1, x_1, \dots, x_m), \\ &\dots \\ \mathbf{b}_{L'} &= (1, 1, \dots, 1, 1, x_1, \dots, x_m). \end{aligned}$$

2. The selection set $\mathcal{B}'_x = \{\mathbf{b}'_1, \dots, \mathbf{b}'_{L'}\}$ is obtained by scrambling all vectors in \mathcal{B}_x .
3. By preceding the vectors in \mathcal{B}'_x with both a 'one' and a 'zero', we obtain the set \mathcal{B}''_x , with $L = 2^r$ elements.
4. The selection set \mathcal{C}_x is obtained by scrambling (precoding) the vectors in \mathcal{B}''_x using the scrambler with polynomial $x + 1$. This embodies the polarity bit principle.
5. The "best" codeword in \mathcal{C}_x is selected.
6. At the receiver end, the codeword is first descrambled using the $x + 1$ polynomial, then after removing the first bit, it is descrambled. The original source word \mathbf{x} is eventually reconstituted by removing the first $(r - 1)$ bits.

All simulations and analyses discussed below assume the above structure, where the selection set consists of pairs of words of opposite disparity. A precise mathematical analysis of the performance of multi-mode codes is, considering its complexity, out of the question. We can either rely on computer simulation to facilitate an understanding of the operation of the coding system or try to define a simple mathematical model, which embraces the essential characteristics of the code and is also analytically tractable. We followed both approaches, and we commence by describing the underlying mathematical model.

10.3.1 The random drawing model

The key characteristic of a multi-mode code is that each source word can be represented by a codeword taken from a set containing L "random" alternatives. As the precise structure of the encoder is extremely difficult to analyze, it is assumed, in our mathematical model, that for each source block \mathbf{x} the channel set \mathcal{C}_x is obtained by *randomly* drawing $L/2$ n -bit words plus their $L/2$ complementary n -bit words. The precise structure of the scrambler is ignored in our model. The "best" word in the set, according to the *minimum running digital sum* (MRDS) criterion, is transmitted. The MRDS criterion ensures that the state space of the encoder, that is, the number of possible *word-end running digital sum* (WRDS) values the encoded sequence may take, is finite. However, if the codewords are relatively long, the number of states and the resulting transition matrix are still too large for a simple mathematical analysis. We therefore truncated the state space by omitting those states that do not contribute significantly to the sum variance.

10.3.2 Transition probabilities of the finite-state machine

The implemented encoder schemes can be simply treated in terms of Markov models. The set of values that WRDS take prior to transmission of a codeword defines a set of states of a finite-state machine. The shorthand notation $Z^{(i)}$ is used to denote both the WRDS at the start of the i th codeword and to refer to the encoder state itself. We commence our analysis with a computation of the state transition probabilities.

Assume the i th codeword starts with RDS $Z^{(i)} = Z'$. Then the multi-mode code can be cast into a Markov chain model whose state transition probabilities matrix, T , is given by

$$T[Z', Z''] = \mathcal{P} \left(Z^{(i+1)} = Z'' \mid Z^{(i)} = Z' \right).$$

We make the following remarks concerning the state transition matrix:

1. For the sake of simplicity, only codes using codewords of even length are considered.
2. It is assumed that at the start of the transmission WRDS is set to +1. As a result, since the codeword length is even, $Z^{(i)} \in \{\pm 1, \pm 3, \dots\}$.
3. For reasons of symmetry, only the probabilities for $Z' > 0$ need to be calculated.
4. The state space was reduced by considering only those states that can be reached from the $Z' = 1$, or the $Z' = -1$, state with probability greater than ϵ , where ϵ is chosen suitably small, say 10^{-6} . Other values of ϵ have been tried without, however, causing significant differences in the results obtained. The remaining states will be termed *principal states*.

We now introduce several notations. If WRDS is positive, then, according to the simple MRDS criterion, the next codeword will be of zero or negative disparity. Therefore, assuming that the encoder occupies state Z' , the set of possible next states is $\mathcal{Z}_{Z'} = \{Z', Z' - 2, \dots, Z' - n\}$. Let $p(d)$ denote the probability of a codeword pair having disparity $+d$ and $-d$. The probability of the *next-state candidate* in a draw being Z^* is

$$p_{Z^*} = \begin{cases} p(|Z^* - Z'|) & \text{if } Z^* \in \mathcal{Z}_{Z'}; \\ 0 & \text{otherwise.} \end{cases} \quad (10.2)$$

The next-state candidate in the j th draw is denoted by Z_j^* , $j = 1, \dots, L'$. According to the MRDS criterion, if the next state is Z'' , then $|Z_j^*| \geq |Z''|$ for all j . The probability that during a draw the next-state candidate is “worse” than Z'' , denoted by $q_{Z''}$, is given by

$$q_{Z''} = \sum_{|Z^*| > |Z''|, Z^* \in \mathcal{Z}_{Z'}} p_{Z^*}.$$

Now, the expression for the transition matrix T is given by

$$T[Z', Z''] = \frac{p_{Z''}}{p_{Z''} + p_{-Z''}} \left[(p_{Z''} + p_{-Z''} + q_{Z''})^{L'} - q_{Z''}^{L'} \right]. \quad (10.3)$$

The transition probabilities for each pair of WRDS states can be numerically determined by invoking (10.3). In order to make the analysis more tractable, those states are removed that can be reached from the $Z' = 1$, or the $Z' = -1$, state with probability less than ϵ . The remaining set of states, the principal states, denoted by $\mathcal{S}_K = \{-K, -K + 2, \dots, K - 2, K\}$, and the truncated transition probability matrix T with elements $t_{i,j}$, $i, j \in \mathcal{S}_K$ can easily be found. Thereafter, the vector of the stationary probabilities, $\boldsymbol{\pi}$ with elements π_i , $i \in \mathcal{S}_K$, is found by solving $\boldsymbol{\pi}T = \boldsymbol{\pi}$. The calculation of the variance of the digital sum at the start of the codewords is now straightforward. The computation of the sum variance within the codewords is quite involved and therefore omitted.

10.3.3 Computational results

The efficiency of dc-balanced codes, E , was defined by (8.66), page 230 as the ratio of the 'redundancy-sum variance' product of the implemented code and that of the maxentropic counterpart. That is

$$E = \frac{\{1 - C(N)\}\sigma_z^2(N)}{\{1 - R\}s^2},$$

where $C(N)$ and $\sigma_z^2(N)$ are the capacity and sum variance of maxentropic dc-balanced sequences with sum variation N . Note that for large sum variation, N , the 'redundancy-sum variance product' of maxentropic (z) sequences is approximately constant (see Eq (8.34), page 213) and equals 0.2326. So that

$$E \approx \frac{0.2326}{\{1 - R\}s^2}.$$

We calculated the sum variance of sequences generated by the random drawing algorithm for selected values of the codeword length and redundancy. The computation of the efficiency E is then straightforward. Figure 10.4 shows the results. The connected points have the same redundancy $(1 - R)$, and the i th point on a curve corresponds to a code having i redundant bits, codeword length $i/(1 - R)$, and selection sets of size 2^i . For comparison purposes, we also plotted the efficiency of the polarity bit code (see Section 8.6.1, page 229). By comparing the efficiency values at the i th point on each curve, it can be seen that these values are approximately the same.

The efficiency of the random coding algorithm is practically independent of the codeword length and is essentially determined by the *number* of redundant bits used. It can be seen that codes with two or three redundant bits are clearly more efficient than the polarity bit code. With an increasing number of redundant bits, however, the efficiency decreases. The decrease in performance, as will be explained in the next section, is due to the shortcomings of the MRDS criterion.

10.3.4 Alternative metrics

The results, plotted in Figure 10.4, reveal that the usage of more than two redundant bits does not lead to improved performance. The reason that the performance decreases with a mounting number of redundant bits can easily be understood. A quick calculation will make it clear that a large selection set contains with great probability at least one zero-disparity word. On the basis of the simple MRDS criterion one of the zero-disparity words is randomly chosen and transmitted. As the sum variance of the full set of zero-disparity codewords equals $(n + 1)/6$, (see Section 8.6.1, page 229) irrespective of the rate of the code, we conclude that the efficiency

will asymptotically approach zero. More sophisticated metrics, which take account of the running digital sum within the codeword, and not only at the end of the word, may result in increased performance.

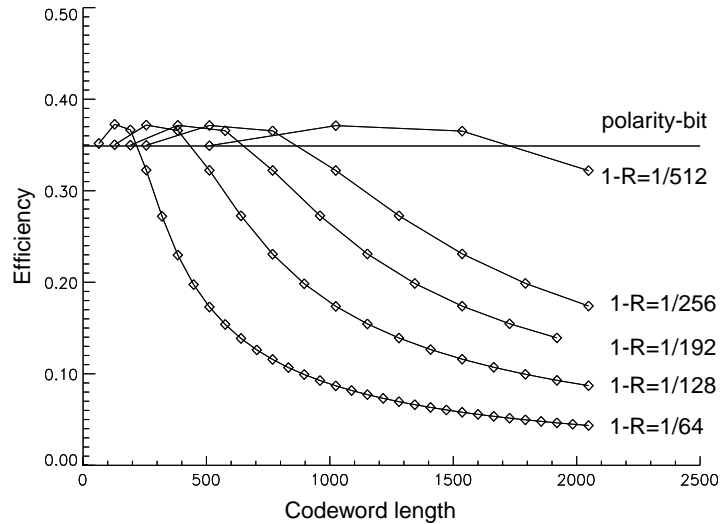


Figure 10.4: Efficiency of random drawing algorithm using the MRDS selection criterion.

In order to describe these more sophisticated metrics we introduce the *squared weight*, w_{sq} , of a codeword, defined as the sum of the squared RDS values at each bit position of the codeword.

The two metrics examined are

1. Modified MRDS (MMRDS) criterion: from the codewords with minimal $|\text{WRDS}|$, the one with minimum w_{sq} is selected.
2. Minimum Squared Weight (MSW) criterion: the codeword of minimal w_{sq} is selected from the selection set, irrespective of the WRDS of the codeword.

Figure 10.5 shows the simulation results obtained for redundancy 1/128. From the simulations, we infer the following characteristics:

- The MRDS method wastes the opportunity offered by the broader selection sets. By properly selecting the codeword from the ones with minimal $|\text{WRDS}|$, the efficiency of the MMRDS scheme tends to unity.
- As indicated by the curve of the MSW criterion, the best codewords do *not* necessarily minimize the $|\text{WRDS}|$. Selecting the codeword with minimal squared weight clearly results in more efficient codes.

Based on the above observations, we searched for a metric that is simple to implement while its efficiency approaches that of the MSW criterion. The outcome is described in the next section.

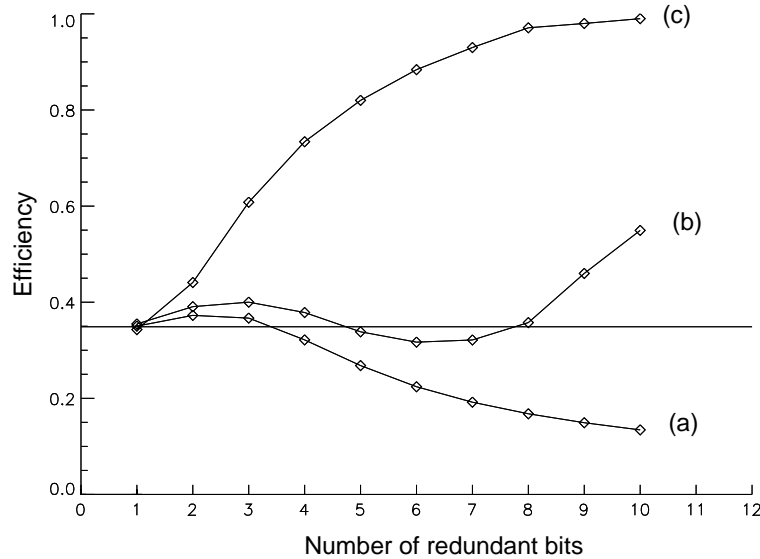


Figure 10.5: Simulation results for the random drawing algorithm. The redundancy is fixed at $1/128$ for all simulations. Three different metrics, (a) MRDS, (b) MMRDS, and (c) MSW, are compared. Simulations of codes with other values of the redundancy produced similar results.

The minimum threshold overrun criterion

Our objective, in this section, is to construct a metric which takes into account the RDS values within the codeword while having a structure that is also easy to implement. The proposed selection scheme, termed *minimum threshold overrun* (MTO) criterion, utilizes the parameter "RDS threshold", denoted by M , $M > 0$. The MTO penalty is simply the number of times the absolute value of the running digital sum within a word is larger than M . As the squaring operation needed for the MSW criterion is avoided, the implementation of the MTO criterion is not more complex than the MRDS method. The codeword with minimum penalty is transmitted. If two or more codewords have the same penalty, one of them is chosen randomly and transmitted. This procedure does not seriously deteriorate the performance as it is fairly improbable that two or more codewords in the selection set have the same penalty value.

Figure 10.6, curve (b), shows simulation results obtained with the MTO criterion. Optimal values of the threshold M were found by trial and error. The MTO criterion is only slightly less efficient than the MSW criterion.

All results shown so far have been obtained by a simulation program of the random drawing algorithm. As a final check we also conducted simulations with a full-fledged implementation using a scrambler with polynomial $x^7 + x + 1$. Experiments with other scrambler polynomials did not reveal substantial differences.

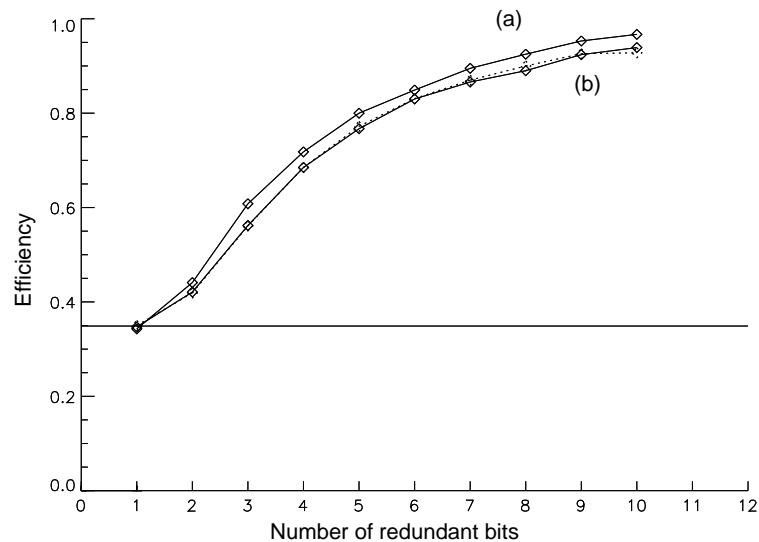


Figure 10.6: Simulation results for the random drawing algorithm having *fixed* redundancy $1/128$ with (a) the MSW criterion and (b) the MTO criterion. The dotted line shows the results obtained for the implemented encoding scheme using a scrambler with polynomial $x^7 + x + 1$.

The dotted curve, Figure 10.6, gives results on the basis of the MTO criterion. The curve shows a nice agreement with results obtained with the random drawing algorithm.

10.4 Evaluation using Hadamard Transform

Copeland & Tezcan [64] described a new method that offers a significant saving in the complexity of the quality evaluation process. Their newly proposed scheme exploits the Fast (Walsh-)Hadamard Transform (FHT). The FHT makes it possible to efficiently generate the disparity of all codewords in the selection set. The entries of the $n \times n$ Hadamard matrix, H_n , can be expressed as

$$h_n[i, j] = (-1)^{\sum_k i_k j_k},$$

where $i_k \in \{0, 1\}$ and $j_k \in \{0, 1\}$ represent the k -th bit in the binary representation of the integers i and j , respectively. This structure leads

to very efficient methods for computing the Hadamard matrix. Let the codeword length be $n = 4$. Then

$$H_4 = \begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & -1 & 1 & -1 \\ 1 & 1 & -1 & -1 \\ 1 & -1 & -1 & 1 \end{bmatrix}.$$

Let the n -bit source word be denoted by \mathbf{x} . The selection set consists of the n (candidate) codewords that can be formed by the transformation (scrambling) of the source word \mathbf{x} with the n columns of the Hadamard matrix. Then the entries v_i of the vector \mathbf{v} given by

$$\mathbf{v} = H_n * \mathbf{x}$$

equal the disparity of the candidate codewords obtained by inverting the bits in \mathbf{x} on the positions, where the entries of the i th column of the Hadamard matrix equals -1. The attractiveness of the FHT stems from the fact that \mathbf{v} can be evaluated with a number of computations that is proportional with $\log n$ instead of n . The encoder transmits that codeword from the selection set having an RDS as close to zero as possible. In addition, the encoder transmits the binary representation of the corresponding column index i to make it possible for the decoder to reconstitute the original source word. We require $\log_2 n$ bits for representing the index so that the rate of the code is

$$R = n / (n + \log_2 n).$$

There are two drawbacks of the above scheme. Firstly, the coding system is very sensitive to error propagation as erroneous reception of the bits representing the column index could easily destroy large portions of the decoded codeword. Secondly, the FHT scheme uses the disparity (and RDS) as quality metric, and, as we have shown above, this will not lead to adequate lf-suppression when the codeword is relatively long. Given the above serious drawbacks, it is not anticipated that the FHT scheme will be readily applied in practice.

10.5 Comparison with alternative methods

It seems appropriate to compare the performance of the guided scrambling method with other suitable coding techniques that can accommodate high-rate codes. The polarity switch encoding method, detailed in Chapter 8, is very suitable for this purpose. Here the basic polarity bit method is extended with a look-ahead (LA) algorithm. The look-ahead algorithm looks ahead p codewords, and evaluates, based on a suitable metric, the

full search tree of 2^p possible choices of the polarity of the codeword in the tree. The tree is evaluated prior to the transmission of each new n -bit codeword. Thus the number of sum variance evaluations are the same for both methods, but note that the 2^p evaluations in the search tree have to be performed every n -bit codeword, while in GS the 2^p evaluations have to be performed every block

As we have seen in this chapter, the spectral performance of any dc-control algorithm depends heavily on the metric used for selecting a codeword. For both encoding schemes we used the sum variance of the new codeword (and the future words in LA) as a metric for the selection of a codeword. We have conducted many simulations of which we will present a few typical results.

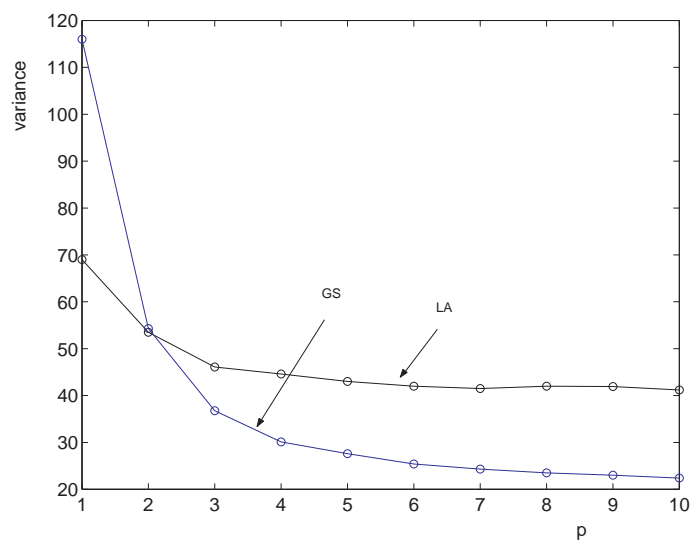


Figure 10.7: Sum variance of sequences encoded by the guided scrambling (GS) method and the look-ahead (LA) polarity switch method as a function of p . The code redundancy is fixed at 1%. The LA method uses a search tree of 2^p branches, while the GS method has a selection set of 2^p codewords.

Figure 10.7 shows the sum variance of the encoded sequence as a function of p . The redundancy of both, LA and GS, coding techniques is fixed at 1%. Figure 10.8 shows the power spectral density $H(f_c) @ f_c = 10^{-4}$ obtained by the two methods. We observe that both methods achieve approximately the same amount of suppression at very low frequencies for $p > 2$. The sum variance of sequences, which are encoded by the guided scrambling method, however, is significantly smaller than that obtained by the look-ahead polarity switch method. We may readily observe the difference in performance of both schemes. In the instance shown, the spectral notch of

the sequences encoded by the GS method is about 50% wider than the one produced by the LA method.

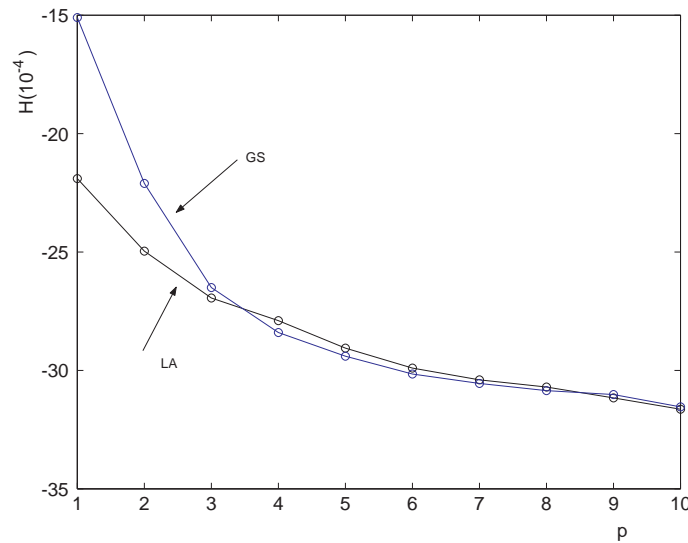


Figure 10.8: Spectral density measured at $f_c = 10^{-4}$ of sequences encoded by the guided scrambling (GS) method and the look-ahead (LA) polarity switch method as a function of p . The code redundancy is fixed at 1%. The LA method uses a search tree of 2^p branches, while the GS method has a selection set of 2^p codewords.

10.6 Weakly constrained codes

Guided scrambling is a typical example of an embodiment of a weakly constrained code. Each source word of length m , $m \gg 1$, is supplemented by r bits, so that the codeword length is $n = r + m$. The rate of the code is $R = m/(r + m)$. The r supplement bits make it possible to generate a selection set of $L = 2^r$ n -sequences. The particular method for generating the selection set is not discussed here; we merely assume that the selection set comprises sufficiently distinct and random words.

Assume we have a channel constraint which limits the channel capacity to C . Then, according to Shannon [296], (see Chapter 2) the number, $N(n)$, of constrained codewords can be approximated by

$$N(n) \approx A2^{nC}, \quad (10.4)$$

where A is a constant. The probability that in L drawings from randomly generated sequences we will not find any sequence that obeys the given constraint is simply

$$p = (1 - p_0)^L, \quad (10.5)$$

where

$$p_0 = \frac{N(n)}{2^n} \approx A2^{(C-1)n}. \quad (10.6)$$

As $L = 2^r$ and $r = (1 - R)n$, we have

$$p = \left(1 - A2^{(C-1)n}\right)^{2^{(1-R)n}}. \quad (10.7)$$

If for simplicity it is assumed that $A2^{(C-1)n} \ll 1$, we have

$$\ln(P) = -A2^{(C-R)n}. \quad (10.8)$$

Clearly, if $R < C$, it is possible to reduce the violation probability p by choosing a sufficiently large codeword length n . The engineering difficulty is that the number of computations grows exponentially with $2^{(1-R)n}$. In particular for low-rate codes the number of computations can easily become prohibitive, and, therefore, the application of the weakly constrained codes is limited to high-rate codes. To illustrate the effectiveness of this idea a worked example of a high-rate $(0, k)$ RLL code will be given in the next subsection. It should be noted, however, that the guided scrambling method is extremely versatile. It can easily be configured to remove all kind of patterns such as the sync or preamble field, or, alternatively, patterns that could flaw the automatic gain control (AGC) or phase information.

10.6.1 Weak $(0, k)$ codes

As an example, we will calculate, in this subsection, for a $(0, k)$ code the parameters A and $C(0, k)$, and substitute them into (10.7) to be able to judge the performance. We start with the computation of the constant A .

The number of self-concatable $(0, k)$ words, $N_c(n)$, of length n , equals the coefficient a_n of the following generating function (see Section 6.6, page 157)

$$\sum a_i x^i = \frac{q(x)}{p(x)} = \frac{x(1 - x^{l+1})(1 - x^{r+1})}{(1 - x)(1 - 2x + x^{k+2})} \quad (10.9)$$

The parameters $l = \lceil k/2 \rceil$ and $r = k - l$ denote the maximum number of 'zeros' with which the words start or end, respectively. For large codeword length n the number of codewords can be approximated by

$$N(n) \approx A\lambda^n, \quad (10.10)$$

where $\lambda = 2^{C(0, k)}$ is the largest real root of the characteristic equation (see [4.16])

$$z^{k+2} - 2z^{k+1} + 1 = 0 \quad (10.11)$$

and the constant A equals (see eq. (6.32), page 158)

$$A = -\lambda \frac{q(1/\lambda)}{p'(1/\lambda)}. \quad (10.12)$$

With the above expressions, we are now in the position to invoke (10.7). Figure 10.9 shows the violation probability, p , that no sequence taken from a selection set of size L of random sequences obeys the $(0, k)$ constraint with the codeword length n as a parameter. The code redundancy is 1%.

We may notice that for longer codewords, and thus larger selection sets, the violation probability goes down more and more rapidly as long as $C(0, k) - R > 0$. In this example, where $R = 0.99$, the violation probability p decreases with mounting codeword length n , if $k \geq 4$ (see Table 4.4, page 60).

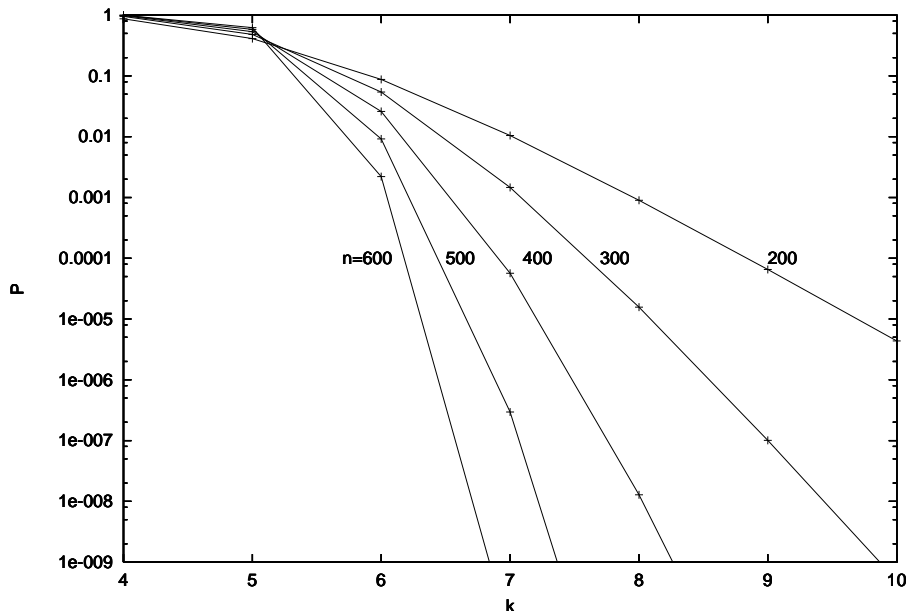


Figure 10.9: Probability, p , that no sequence of a sample of size $L = 2^{n/100}$ of random sequences of length n satisfies the $(0, k)$ constraint versus k with the codeword length n as a parameter. The code redundancy $1 - R$ is 1%.

Let us examine the example, where the codeword length is $n = 400$. Then the size of the selection set is $L = 2^{400/100} = 16$. Figure 10.9 shows that with very small probability, $\ll 10^{-9}$, the encoder will fail to find a viable codeword that satisfies the $k = 9$ constraint. The $k = 8$ constraint is violated with probability $2 \cdot 10^{-8}$. By contrast, the conventional "strict-to-the-book" implementation of a $(0, 9)$ code, see Section 5.6.3, page 115, requires a redundancy of $1 - R = 1/25$ —four times the redundancy of the weakly constrained

code— to guarantee the same $k = 9$ constraint. A comparison between the properties of maxentropic $(0, k)$ sequences and those of sequences generated under the rules of guided scrambling have been conducted by Kunisa [211, 210].

The above computational results are very exciting. They show, among others, that with a relatively small redundancy, $1 - R = 0.01$ and selection set of size, $L = 16$, we can, on the average, achieve an extremely small probability of violation errors for, say, $k \geq 8$. There is no difficulty to cast the above code into silicon. But, the drawback of weakly constrained codes is, of course, that they are sensitive for series of specific, worst case, source data. A high-rate $(0, k)$ code, serving as a 'safety net' that guarantees for example $k = 11$, combined with the guided scrambling algorithm might provide a sound trade-off between redundancy, worst case performance, and average performance. Results are given in [346].

An alternative method for generating weakly constrained RLL $(0, k)$ codes was proposed by Ming *et al.* [250]. The encoding is done in two steps. In the first step, the encoder inserts (stuffs) a 'one' after every string of k consecutive 'zero's, which leads to a string of variable length. This first encoding step, by the way, is a traditional method in communications systems for constraining the maximum runlength [217]. In the second step, the variable-length string so obtained is made synchronous (i.e., it has the required feature of fixed output block length) by adding a certain number of dummy bits. The number of dummy bits added depends on the number of bit insertions made in the first step, where the total number of bits (stuffed bits plus dummy bits), q , added to the source sequence is fixed. As the proposed coding scheme cannot guarantee the fixed-length requirement, as at most q bit stuffings can be accommodated, without disobeying the maximum runlength constraint, this leads to a class of weakly constrained codes. The authors analyzed the codes, and found that the codes offer results which are better or comparable to those of currently available 'rigid' $(0, k)$ codes, however, at the cost of encoder failure with arbitrarily low probability.

Chapter 11

Dc-free RLL Codes

11.1 Introduction

This text has reviewed the development of several techniques for the design of codes whose output meets prescribed runlength constraints and we have investigated the theoretical properties and practical implementations of dc-balanced sequences. This chapter reviews binary sequences with combined constraints on the runlengths and dc-unbalance. A runlength-limited sequence whose running digital sum is bounded, is characterized by three parameters d , k , and N , where d and k are the runlength parameters described in Chapter 4 and the running digital sum z_i (RDS) at any digit position in the coded sequence is bounded by $N = \max_i |\{z_i\}|$ units (see Chapter 8). A little thought will make it clear that a constraint on the running digital sum automatically sets a bound to the maximum runlength, namely $k \leq N - 2$.

In the following sections of this chapter, the properties of maxentropic dc-free RLL (DCRLL) sequences are first considered. Thereafter, in Section 11.4, it is shown how practical codes can be devised that satisfy the given channel constraints. It will be shown that industry-standard RLL codes can be supplemented by a simple mechanism with which the lf-components of the generated sequences can be suppressed. In the remaining part of this chapter, we will study the construction and performance of EFM-like codes. We start in the next section, with the computation of the capacity of DCRLL sequences.

11.2 Capacity of DCRLL codes

Using the techniques discussed in Chapter 8, it is now fairly straightforward to compute the capacity of sequences with a constraint on both the runlength and the number of sum values the sequence assumes. We follow

the approach given by Norris & Bloomberg [264] whose early work on this topic coincides with the study provided in Chapter 2.

The capacity of a runlength-limited sequence with a bounded running digital sum is characterized by three parameters d , k , and N , and will be denoted by $C(d, k, N)$.

Chapter 4 provides an expression of the capacity $C(d, k, \infty)$ of RLL sequences:

$$C(d, k, \infty) = \log_2 \lambda_{dk}, \quad (11.1)$$

where λ_{dk} is the largest real root of the characteristic equation

$$z^{k+1} - z^{k-d} - z^{k-d-1} - \dots - z - 1 = 0. \quad (11.2)$$

The capacity $C(0, N-2, N)$ of a sequence that assumes a maximum of N sum values is computed in Chapter 8, and it is given by

$$C(0, N-2, N) = 1 + \log_2 \cos \frac{\pi}{N+1}. \quad (11.3)$$

Obviously,

$$C(d, k, N) \leq \min\{C(d, k, \infty), C(0, N-2, N)\}. \quad (11.4)$$

Let the number of distinct sequences of length n with upward and downward directed transitions and running digital sum z , $1 \leq z \leq N$, be denoted by $N_u(z, n)$ and $N_d(z, n)$, respectively. We obtain

$$\begin{aligned} N_d(z, n) &= \sum_{j=d+1}^{k+1} N_u(z-j, n-j) \\ N_u(z, n) &= \sum_{j=d+1}^{k+1} N_d(z+j, n-j). \end{aligned} \quad (11.5)$$

For sufficiently large sequence length n , symmetry about $z = 0$ yields

$$N_d(z, n) = N_u(-z, n). \quad (11.6)$$

The above relations define a set of linear difference equations which can be solved by assuming the solution

$$N_u(z, n) = u(z)\lambda^n$$

and

$$N_d(z, n) = d(z)\lambda^n, \quad 1 \leq z \leq N.$$

Writing out gives

$$d(z) = \sum_{j=d+1}^{k+1} d(-z+j)\lambda^{-j}, \quad 1 \leq z \leq N. \quad (11.7)$$

This is a set of N homogeneous linear equations. We define the matrix $D(\lambda)$ with elements

$$d_{ij} = \lambda^{-(i+j-N-1)} f(i+j-N-1), \quad 1 \leq i, j \leq N,$$

where

$$f(p) = \begin{cases} 1, & \text{if } d+1 \leq p \leq k+1 \\ 0, & \text{otherwise.} \end{cases}$$

The capacity of the constrained channel, given the parameters N , d , and k , is found by

$$C(d, k, N) = \log_2 \lambda_{\max},$$

where λ_{\max} is the greatest real root of

$$\det[D(\lambda) - I] = 0.$$

In the limit $N \rightarrow \infty$, all RDS states are equally likely and the set of linear equations degenerates to one equation:

$$\sum_{i=d+1}^{k+1} \lambda^{-i} - 1 = 0.$$

This equation coincides with (4.16), page 59. Table 11.1 shows the results of computations for various values of digital sum variation and runlength parameter. The following examples may serve to illustrate the theory.

Example 11.1 Let $(d, k, N) = (0, 1, 3)$. The characteristic equation is

$$\begin{aligned} \det[D(\lambda) - I] &= \begin{vmatrix} -1 & 0 & 0 \\ 0 & -1 & \lambda^{-1} \\ 0 & \lambda^{-1} & \lambda^{-2} - 1 \end{vmatrix} \\ &= \lambda^2 - 2 = 0. \end{aligned}$$

The largest root is $\sqrt{2}$, so that the capacity is $C(0, 1, 3) = 1/2$. The bi-phase code, the details of which are treated in the previous chapters, is an interesting example of a code embodiment that achieves a rate equal to the capacity for the prescribed channel constraints.

Example 11.2 Consider the set of constraints $(d, k, N) = (1, 3, 7)$. The characteristic equation is

$$\begin{aligned} \det[D(\lambda) - I] &= \lambda^8 - \lambda^6 - 3\lambda^4 + \lambda^2 + 2 \\ &= (\lambda^2 + 1)^2(\lambda^2 - 1)(\lambda^2 - 2) = 0. \end{aligned}$$

The largest root is $\sqrt{2}$, so that the capacity is $C(1, 3, 7) = 1/2$. The Zero-Modulation code to be discussed shortly is a rate = 1/2 code which complies with the given runlength parameters and maximum sum variation.

Table 11.1: Capacity of dc-balanced runlength-limited sequences.

d	k	$N = 5$	$N = 6$	$N = 7$	$N = 8$	$N = 9$
0	1	.6358	.6551	.6662	.6731	.6778
0	2	.7664	.8032	.8244	.8378	.8468
0	3	.7925	.8416	.8704	.8887	.9012
0	4		.8495	.8832	.9048	.9196
0	5			.8858	.9094	.9256
0	6				.9103	.9273
0	7					.9276
1	2	.3471	.3705	.3822	.3889	.3931
1	3	.4248	.4746	.5000	.5145	.5237
1	4		.5018	.5390	.5608	.5746
1	5			.5497	.5772	.5947
1	6				.5816	.6020
1	7					.6039
2	3	.2028	.2457	.2625	.2709	.2757
2	4		.3089	.3471	.3666	.3777
2	5			.3723	.4024	.4199
2	6				.4135	.4366
2	7					.4418
3	4		.1568	.1903	.2035	.2101
3	5			.2434	.2744	.2902
3	6				.2972	.3224
3	7					.3333

11.3 Spectrum of maxentropic DCRLL sequences

The power spectral density function of maxentropic sequences can, in principle, be found using a straightforward generalization of the techniques described in Chapter 3. The presentation of the finite-state machine underlying the specified constraints is somewhat awkward, since the number of states, about $N \times k$, required to describe the machine is fairly large. Kerpez, [202] presented a description of the combined (d, k) and N constraint in terms of a variable length graph and its adjacency matrix/indexadjacency matrix that requires a relatively small number, $N - 1 - d$, of states.

Let the DCRLL message be denoted by $X = \{x_0, x_1, \dots\}$, $x_i \in \{-1, 1\}$. The RLL message is assumed to be composed of runlengths of lengths T_i , $i = 0, 1, \dots$, taken from the set of allowed runlengths $dk = \{d+1, \dots, k+1\}$.

As the sequence X has limited digital sum variation (DSV) we have

$$\left| \sum_{i=0}^j x_i \right| \leq c,$$

where $N = 2c + 1$. The above constraint can be described in terms of the runlengths. The sequence X is composed of a cascade of runlengths T_i whose symbols have alternate polarity. Transitions of the polarity of the sequence X , i.e. instants where $x_i \neq x_{i+1}$, occur therefore at $t_j = \sum_{i=0}^j T_i$. We simply find

$$\left| \sum_{i=0}^{t_j} x_i \right| = \left| \sum_{i=1}^j (-1)^i T_i \right| = |U_j|,$$

where the sequence $U_j = T_j - U_{j-1}$, $U_0 = 0$. In other words the DSV constraint is equivalent to

$$|U_j| \leq c \text{ for all } j.$$

Thus a sequence satisfies the $(d, k, N = 2c + 1)$ constraint if and only if the sequence of runlengths $\{T_j\}$ satisfies, for all j ,

$$d + 1 \leq T_j \leq k + 1 \quad (\leq 2c) \tag{11.8}$$

and

$$d + 1 - c \leq U_j \leq c. \tag{11.9}$$

The general form of the adjacency matrix D for the $(d, k, N = 2c + 1)$ constraint, derived from (11.8) and (11.9), has a regular structure. The matrix has size $(N - 1 - d) \times (N - 1 - d)$, and is constant on the anti-diagonals. If the value of k is non-trivial, i.e. $k + 1 \leq N - 1$, the lower right of the diagonal of the matrix is zero,

$$D(z) = \begin{bmatrix} 0 & 0 & 0 & \dots & 0 & \dots & 0 & z^{-d-1} \\ 0 & 0 & 0 & \dots & 0 & \dots & z^{-d-1} & z^{-d-2} \\ \vdots & & & & & & & \vdots \\ 0 & z^{-d-1} & z^{-d-2} & \dots & z^{-k} & \dots & & 0 \\ z^{-d-1} & z^{-d-2} & z^{-d-3} & \dots & z^{-k} & \dots & 0 & 0 \end{bmatrix}, \tag{11.10}$$

while in the case of $k = N - 2$, the lower right corner is filled

$$D(z) = \begin{bmatrix} 0 & 0 & 0 & \dots & 0 & z^{-d-1} \\ 0 & 0 & 0 & \dots & z^{-d-1} & z^{-d-2} \\ \vdots & & & & & \vdots \\ 0 & z^{-d-1} & z^{-d-2} & \dots & & d^{-2c+1} \\ z^{-d-1} & z^{-d-2} & z^{-d-3} & \dots & z^{-2c+1} & z^{-2c} \end{bmatrix}. \tag{11.11}$$

Using the above adjacency matrix it is now quite straightforward to compute the capacity and the spectrum of the maxentropic sequence for large values of N and k . Figure 11.1 shows the power spectral density function of maxentropic sequences with $N = 7$, $d = 1$ and the maximum runlength k as a parameter. Apparently, the influence of the maximum runlength parameter is drastic. Most noticeable is the fact that the curves become more peaked with decreasing maximum runlength parameter k .

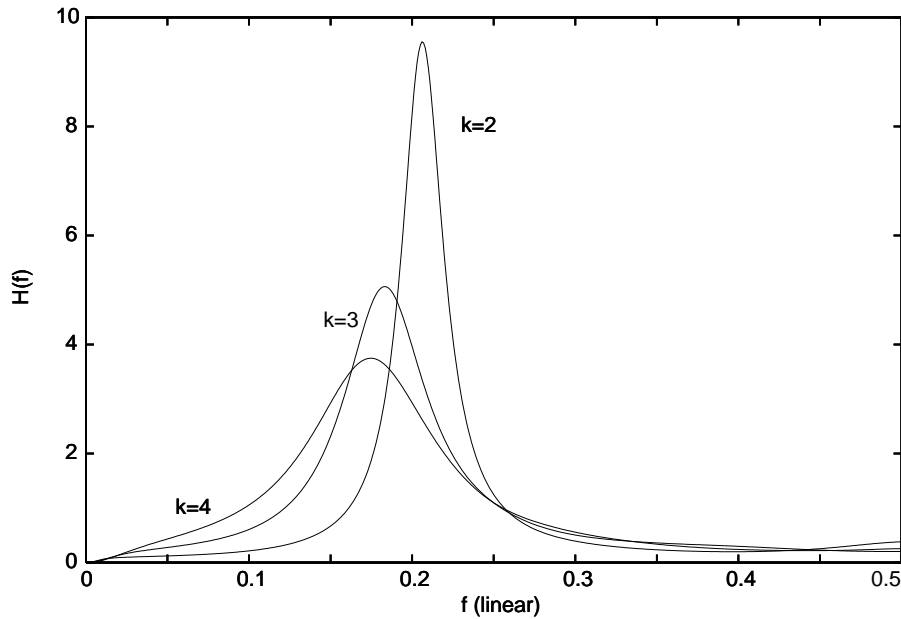


Figure 11.1: Power spectral density function of a maxentropic dc-balanced, runlength-limited sequence. Digital sum variation $N = 7$, and runlength parameters $d = 1$ and $k = 2, 3, 4$.

Figure 11.2 shows the spectrum for the parameters $d = 2$, $k = 10$ and $N = 12, 15, 21$, and $N = 33$ with a logarithmic frequency-axis and a vertical $H(f)$ (dB) axis, where a dB is defined by $10 \log_{10} H(f)$. The choice of the log axes clearly shows the parabolic relationship,

$$H(f) \approx af^2,$$

between power and frequency in the low frequency range. The low-frequency power increases with 6 dB per octave (or 20 dB per decade) frequency increase.

As in the instance of 'pure' dc-free codes, we need a sound yardstick for measuring the low-frequency properties of DCRLL sequences. In Chapter 8 the spectral width is quantified by a parameter called *cut-off frequency* ω_0 . As discussed in Chapter 8 the sum variance of the sequence is closely related

to the cut-off frequency. In a similar fashion, Braun [43] defined the cut-off frequency of DCRLI sequences, denoted by ω_0 , by

$$H(\omega_0) = \frac{H_0(d, k)}{2}, \tag{11.12}$$

where $H_0(d, k)$ denotes the spectral density at zero frequency of the maxentropic (d, k) constrained sequence. The value of $H_0(d, k)$ can be computed with (4.35) at page 66. For $d = 2$ and $k = 10$, the parameters used in Figure 11.2, we find $H_0(d, k) = 0.835 (= -0.8 \text{ dB})$.

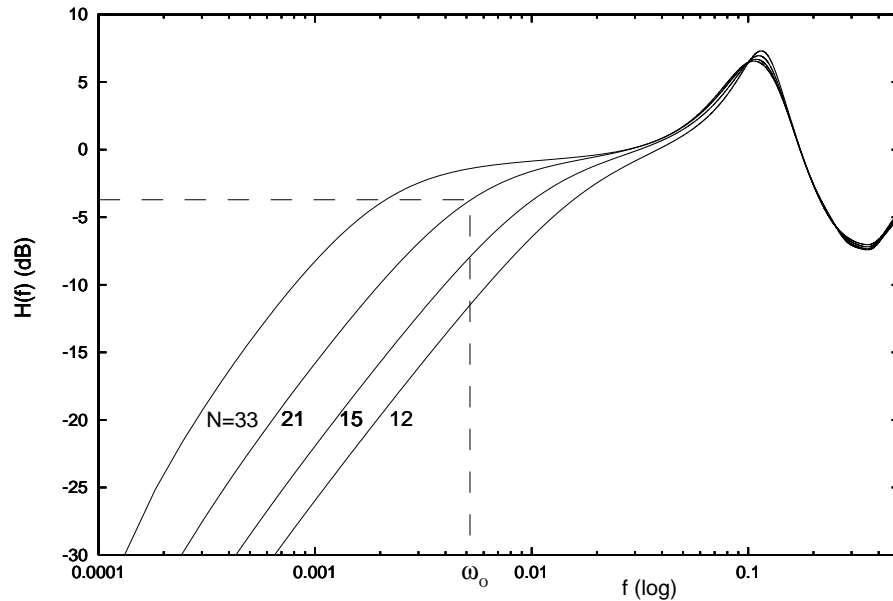


Figure 11.2: Power spectral density function of a maxentropic dc-balanced RLL sequence (logarithmic axes). Runlength parameters $d = 2$ and $k = 10$ with N as a parameter. By way of example, the cut-off frequency ω_0 is shown for $d = 2, k = 10$, and $N = 21$.

Braun also studied the relationship between redundancy and cut-off frequency. He defined the *extra redundancy* as the difference between the capacities of the pure RLL channel and the DCRLI channel, or

$$\rho(d, k, N) = C(d, k, \infty) - C(d, k, N). \tag{11.13}$$

The parameter $\rho(d, k, N)$ quantifies the redundancy that results from the additional constraint on the running digital sum N . Braun showed that maxentropic sequences have the property that there is, in good approximation, a linear relationship between cut-off frequency ω_0 and the extra redundancy ρ . The relationship is given by

$$\omega_0 \approx \rho(d, k, N) \frac{2 \ln 2}{\frac{\pi^2}{6} - 1}, \quad N \gg 1. \tag{11.14}$$

The constant of proportionality between cut-off frequency and extra redundancy is independent of the d , k , and N constraints. The constant was derived for pure RDS constrained sequences (see Chapter 8), and is also valid if in addition d and k constraints are imposed. Computer simulations revealed that the above relationship appears to be accurate to within 5% for $N > 20$.

In the next section, we will discuss various design methods that emerged in the literature.

11.4 Examples of DCRL codes

The design of any constrained code can, at least in principle, be systematically accomplished by the design techniques discussed in the previous chapters of this book. Unfortunately, the design of a DCRL code whose rate is close to the Shannon capacity of the constrained channel, is severely hampered by the large number of states of the finite-state machine which models the channel constraints at hand. The large number of states of the underlying FSM, can, at least in principle, be handled by buying a sufficiently large computer, but the insight required is too easily lost. The design of DCRL codes is therefore (still) the province of a plurality of *ad hoc* methods, for example [324, 316, 22, 221]. Basically there are four systematic design approaches that emerged in the literature.

The first method uses the ACH algorithm to design an RLL code. As discussed in Chapter 7, in the final stage of the ACH algorithm we end up with a graph with the property that from any state of the graph there are at least 2^m (m is assumed to be the source word length) outgoing edges. There are (hopefully) states with a larger number of outgoing edges. These *surplus edges* are used as alternative codewords that can be used for dc-control. The rate 8/16, (2,10) EFMPPlus code, to be discussed in Section 11.5.2, is an example of a DCRL code used in practice (DVD) that was designed according to these guidelines.

The second systematic method, which was first developed by Lin & Liu [222] and later generalized by Abdel-Khaffar & Weber [1], uses a simple block code plus a number of merging bits for concatenating the words. The number of merging bits is chosen in such a way that there are at least two alternative channel representations with different weight parities. The merging bits are chosen after observing the dk constraint and disparity of the upcoming codeword.

In the third method, dc-control is effectuated by multiplexing the source data or the encoded data with dc-control bits. A given, state-of-the-art, RLL code, for example the rate 2/3, (1,7) code, is used to generate RLL sequences. The sequences generated under the coding rules of said code

are multiplexed with channel bits for minimizing the low-frequency components, the *dc-control*. The user data or alternatively the encoded data are partitioned into segments of ν bits. Basically, there are two approaches with which a (d, k) (or other constrained) encoder can be extended with multiplexed dc-control, namely at source data level or at channel data level. The two multiplex approaches of dc-control have various distinct features.

Between two consecutive ν -bit segments β dc-control bits are inserted, and the β dc-control bits, in turn, are chosen to minimize the low-frequency components. In the experimental phase, we have the freedom to select the parameters ν and β such that the required dc-suppression is reached. There is, in other words, no need to redesign the constituent RLL code.

The fourth construction uses a new technique developed by Fair *et al.* [76], called *Guided Scrambling*. In Guided Scrambling, a selection set of 2^β alternative RLL sequences is generated, and the encoder selects that sequence that "best" matches the spectral requirements. In order to create such a selection set, we need β redundant bits per block of ν data bits. As in the second method, time multiplex, we have the flexibility to choose the parameters ν and β . The results are excellent, and more results are presented in Chapter 10.

The success of the design method depends on various factors such as, for example, how much lf-suppression is required. In most of the practical cases that the author encountered an extra redundancy for the dc-control of 2-3% was sufficient to yield the required dc-suppression. In that instance, codes using multiplexing methods offer an excellent performance and design flexibility.

In the next subsections we will describe two methods for dc-control, while the third method, Guided Scrambling, is discussed in Chapter 10, page 257.

11.4.1 ACH-algorithm-based DCRLI codes

A simple and effective method for designing DCRLI codes is based on the application of the ACH algorithm (or other schemes for designing encoder graphs). Define the usual parameters such as m , n , d , and k , and apply the ACH algorithm. Then we will find an encoder graph with the property that from each encoder state there are at least 2^m code words. In some instances there are more than the required code words available, so that the surplus words can be used as alternative channel representations for minimizing the spectral content at the low frequency end. A very simple example may illustrate this.

Example 11.3 Let $d = 2$, $k = \infty$, $m = 2$, and $n = 4$. Baldwin [22], see also Section 7.6, page 181, presented a two-state ($d=2$) RLL encoder, where the

surplus codewords are used as alternative channel representations. For the case $n = 4$, two words can be combined to form an alternative pair. Pairing of words, such that it has the greatest impact on the spectral control, is not straightforward. Here we opted to combine the codewords '1000' and '1001'.

i	$h(i, 1)$	$g(i, 1)$	$h(i, 2)$	$g(i, 2)$
0	0000	1	0100	1
1	0000	2	0100	2
2	0010	1	100x	1
3	0001	1	1000	2

Then we create the possibility to set or not set a transition in the corresponding NRZ sequence, so that we can invert (or not) the sequence following that codeword. In the Table, the word '100x' means that the encoder may choose 'x' to be '0' or '1', i.e., choose between the codewords '1000' and '1001', for minimizing the lf content.

Note that the above encoder does not guarantee the dc-control, as it is not guaranteed that the encoder will ever transmit input word '2' while in State 2. It is not difficult to generalize the concept demonstrated in the above example to other values of d and n , see, for example [197] for a rate $8/12$, $d = 1$ code. The EFMPPlus code ($d=2$), discussed in Section 11.5.2, page 296, is a byte-oriented dk -constrained code.

In the next section, we will present a description of an alternative design method, where dc-control bits are multiplexed with the user or channel data.

11.4.2 Dc-control: data level versus channel level

Assume the d and k constraints are given and that an efficient (d, k) code has been found in the literature or, alternatively, constructed using the various methods offered in this book. A straightforward method for extending a standard (d, k) code with dc-suppression is to add (or stuff) redundant bits, whose values are chosen to reduce the power density at the low-frequency end. The redundant bits are usually called *dc-control bits*. Essentially, there are two approaches with which a (d, k) (or other constrained) encoder can be extended with multiplexed dc-control, namely at (a) *source data* level or at (b) *channel data* level.

Multiplexing at either level is shown in Figure 11.3. Between segments of ν source data or between segments of ν encoded data β dc-control bits are inserted. In both multiplex formats, the β dc-control bits are chosen to minimize the low-frequency components of the channel sequence generated. This can be accomplished by tallying the running digital sum at the end of each candidate segment. The encoder transmits that candidate segment

whose RDS is closest to zero. At the receiver site, the added dc-control bits, either at data or channel level, can easily be skipped by the decoder.

The two multiplex approaches of dc-control have various distinct features. The β dc-control bits can be freely chosen if they are multiplexed at source data level. Then the encoder has 2^β possible sequences to be tried. If, on the other hand, the dc-control bits are multiplexed with the (d, k) sequence, the multiplexed sequence so generated has to obey the (d, k) constraints in force, and as a result the number of candidate sequences to be tried is less than 2^β . For the dc-control to be effective under all worst case circumstances, it must guarantee that an (almost) entire segment of ν modulated data bits can be inverted or not. We can easily verify that if the dc-control bits are multiplexed with the (d, k) sequence, that in order to guarantee said worst case performance, we require at least $\beta = d + 1$ dc-control bits and the maximum runlength at the segment boundaries will increase from k to $k + 1$. Similar methods have been proposed by Odaka [266], Coppersmith & Kitchens [65], Patel [276], and Ino [169]. Coene [60] proposed to employ a *combi* code, which alternates between a first and a second RLL code. The first code has a codeword length n , while the second code has a codeword length $n + \beta$. Then the extra β channel bits are part of the second RLL code, which can then be optimized for dc-control.

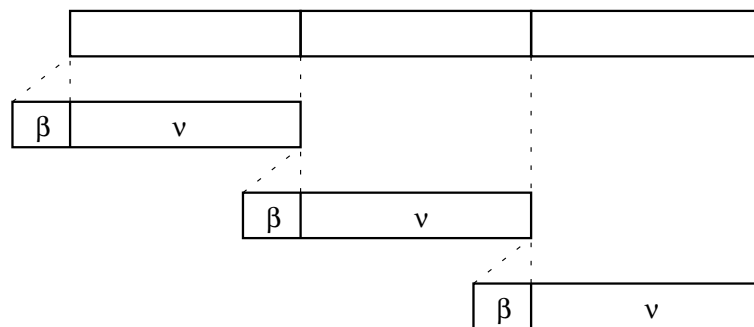


Figure 11.3: The user data or encoded data are partitioned into segments of ν bits, and between two consecutive ν -bit segments β dc-control bits are inserted.

When, on the other hand, dc-control bits are inserted at source level, see Moriyama [256], the matter of worst case performance is much more involved. The encoded segments are both a function of the source data and the encoder state at the start of the segment. It is therefore not recommended to use an industry-standard (d, k) code. A possible solution, using the *parity preserving word assignment*, will be discussed in the next section.

11.4.3 Codes with parity preserving word assignment

In order to make it possible to efficiently control the dc-content in the source data level mode, we have made the assignment between source words and codewords in such a way that the *parity* of both source word and its assigned codeword are the same. The parity, P , of an n -bit word (x_1, \dots, x_n) , $x_i \in \{0, 1\}$, (either source or codewords) is defined by

$$P = \sum_{i=1}^n x_i \bmod 2.$$

In other words, if the source word has an even (or odd) number of 'one's then its channel representation also has an even (or odd) number of 'one's. A code with a *parity preserving* assignment has the virtue that when it is used in conjunction with dc-control bits at data level that setting an even (or odd) number of 'one's at data level will result in an even (or odd) number of 'one's at code level. This leads, as we will demonstrate shortly, to an efficient dc-control.

Table 11.2: Variable-length synchronous rate 2/3, $(1, \infty)$ code with parity preserving assignment.

<i>Data</i>		<i>Code</i>
00	← →	000
01	← →	010
10	← →	100
1100	← →	001010
1101	← →	001000
1110	← →	101010
1111	← →	101000

The variable length rate 2/3, $(1, \infty)$ code shown in Table 7.10, page 168, can be rewritten to comply with the parity preserving property. The result is shown in Table 11.2. It can easily be verified that indeed the assignment is parity preserving. The variable length rate 2/3, $(1, 8)$ code shown in Table 7.11, page 168, can also be rewritten in order to comply with the parity preserving assignment. The list is omitted for space reasons.

The variable length rate 1/2, $(2, 7)$ code, shown in Table 7.9, page 167, can be rewritten. The result is listed in Table 11.3. A parity preserving assignment of a rate 2/3, $(1, 8)$ code, first presented by Kahlman & Imink [181], is based on the look-ahead rate 2/3, $(1, 7)$ code described in Section 7.4.1, page 171. Table 11.4, the main table, and Tables 11.5 and 11.6, the two substitute tables, show the encoding rules of the new code

parity preserving code. A similar look-ahead rate $2/3$, $(1,7)$ parity preserving code was developed by Kahlman *et al.* for the BluRay Disc System [182, 260]. The full coding table of the code consists of a main table and two substitute tables instead of a single substitute table. It can easily be verified that the assignment is indeed parity preserving.

Table 11.3: Variable-length synchronous rate $1/2$, $(2,7)$ code.

<i>Data</i>		<i>Code</i>
10	$\leftarrow \rightarrow$	0100
01	$\leftarrow \rightarrow$	1000
001	$\leftarrow \rightarrow$	001000
000	$\leftarrow \rightarrow$	100100
111	$\leftarrow \rightarrow$	000100
1101	$\leftarrow \rightarrow$	00001000
1100	$\leftarrow \rightarrow$	00100100

Table 11.4: Basic coding table parity preserving $(1,8)$ code.

<i>Data</i>	<i>Code</i>
00	101
01	100
10	001
11	000

Table 11.5: Substituting Coding Table I parity preserving $(1,8)$ code.

<i>Data</i>	<i>Code</i>
00.00	100.010
00.01	101.010
10.00	000.010
10.01	001.010

The code was found by trial and error, as no approach is (yet) available for systematically constructing codes with a parity preserve word assignment. The systematic design of RLL codes with parity preserving word assignment is, a challenging task. The above examples show that it is indeed

possible and that such codes offer a better performance than their counterparts. Block codes are by their virtue of simplicity good candidates, but the complexity issue will hamper their design. Variable length synchronous codes seem to be promising candidates. It is not (yet) clear how we can efficiently design parity preserving codes with the ACH algorithm.

Table 11.6: Substituting coding table II parity preserving (1,8) code.

<i>Data</i>	<i>Code</i>
11.11.11	000.010.010
11.11.10	001.010.010
01.11.10	101.010.010
01.11.11	100.010.010

Figure 11.4: Performance comparison of dc-control at data and channel level. As a comparison the performance of maxentropic DCRL sequence has been plotted.

Performance comparison

The difference between the quality of the alternative dc-control methods has been assessed by Wang *et al.* [332]. The power density measured at a relatively low channel frequency, $f_c/1000$, was used as a quality criterion. Computer programs have been written for simulating the two alternative coding schemes, where the dc-control bits are multiplexed at source or at

channel level, respectively. The code for the channel-level multiplex is the standard, rate $2/3$, $(1,7)$ code, while the source-level multiplex is the parity preserving, rate $2/3$, $(1,8)$ code described in the previous section. He observed that the parity preserving code performs 2 dB better than the standard rate $2/3$, $(1,7)$ code used with channel-level multiplex in the range of dc-control redundancy of 1-4%. Figure 11.4 shows results of computations. As we can observe there is quite some room for improvement with respect to the performance of maxentropic sequences.

11.4.4 Zero Modulation

Zero Modulation (ZM) was designed by Patel [274, 273] for a rotary-head storage system (IBM 3850). The clever idea in ZM is to slightly modify the MFM code so that it becomes dc-balanced. It seems appropriate at this point to reconsider the MFM code (see Chapter 5).

MFM, an $R = 1/2$, $(d = 1, k = 3)$ code, is a simple block code with code-words of length $n = 2$. A simple merging rule when the NRZI notation is employed. The coding rules are shown in Table 11.7. The symbol indicated with 'x' is set to 'zero' if the preceding symbol is 'one' else it is set to 'one'. It can be verified that this construction yields a maximum runlength of $k = 3$.

Table 11.7: Coding rules MFM code.

<i>Source</i>	<i>Output</i>
0	<i>x</i> 0
1	01

By allowing a larger maximum runlength it is possible to create a degree of freedom that provides the opportunity to balance the encoded sequence. Both the ZM code and the Modified Squared code, to be considered in the subsequent section, operate according to this principle. So we have the means to modify some sequences, but what sequences have to be modified? To answer this question we must examine the MFM encoding process in greater detail.

Any source stream may be considered a series of sequences of two types:

(a) 011110 t_n 'one's bounded by 'zero's, $t_n \geq 0$ and

(b) 111111 t_m 'one's.

Under MFM coding rules (change-of-state encoder included), sequence of type (a) with t_n even and non-zero have a non-zero dc-balance which, upon concatenation with interleaving type (b), t_m even sequences, can grow indefinitely. All other sequences, type (a) with t_n odd or zero or type (b), have zero dc-balance. Reflection on the MFM waveforms will reveal why this is so. In the sequences with non-zero dc-balance, ZM encodes the 'zero's in the MFM manner the 'one's, however, are encoded as though they were 'zero's but with alternate transitions deleted. For example, a type (a) source sequence 011110 would be encoded according to MFM rules as x0 01 01 01 01 00 (NRZI notation) but according to ZM rules is encoded as x0 10 00 10 00 10 (NRZI notation). The difficulty with the implementation of this coding scheme is that one has to look for sequences of a certain type. In principle, therefore, infinite look-forward is required to identify the sequence boundaries and to provide the ZM modified sequence if, indeed, t_n turns out to be even. In a practical realization, the amount of memory required for look-ahead can be limited to f bits, where f is a positive integer, by adding a small amount of redundancy by inserting a parity/ indexparity at the end of every section of f source bits. A practical value of f is 128. It is relevant to note that, though the encoder requires infinite look-back and look-forward, the decoder window of the sliding-block decoder is confined to six channel bits, thus guaranteeing a limited error propagation. Karabed & Siegel [188] presented a 100% efficient sliding-block code with the same parameters that can be encoded with a finite-state machine encoder.

11.4.5 Miller-Miller code

Mallinson and Miller [228, 248, 249] published a code, called *Miller-Miller* or M^2 code, which possesses the following properties: $d = 1$, $k = 5$ and $N = 7$. Thus, the maximum runlength has increased with respect to conventional MFM and ZM. In M^2 , as in ZM, the basic MFM code is modified so that it becomes dc-balanced. Again, this is accomplished by modifying the sequences which have non-zero dc-balance. To limit the look-forward operation, however, the modifications are introduced only at the end of the sequences. In the sequences with non-zero dc-balance, M^2 encodes all the source 'one's, apart from the last, in the standard MFM manner; the final 'one' is simply ignored. As a result, the maximum runlength parameter is increased to $k = 5$. The suppression of a 'one' bit transition may be recognized by the failure of a transition to occur in less than five clock periods of a previous transition. Thus, the suppressed data transitions can be detected in the receiver by inspecting no more than five successive channel bits thus limiting the propagation of errors. An alternative of the M^2 code was given by Isozaki [171], who constructed a byte oriented rate 8/16, DCRL (1,4) code. Lin & Liu [222] published an alternative to M^2 , with the parameters

$k = 5$ and $N = 21$. This M^2 alternative uses 16-bit codewords, which include 3-bit merging words. The merging words are operated, as in EFM, to preserve the d and k constraint, and to minimize the lf content. The 3-bit merging words guarantee an effective dc-control.

11.5 EFM revisited

In this section, we will take a closer look at EFM and EFMPlus used in the CD and DVD, respectively. We shall also offer alternatives to EFM coding, so-called EFM-like codes.

11.5.1 EFM

As discussed in Chapter 5 the main parameters of EFM are $d = 2$, $k = 10$, and rate $R = 8/17$. Detailed information can be found in the patent issued to Immink & Ogawa [165]. The 8-bit source data is translated into a 14-bit d -constrained word. The 14-bit words are concatenated with 3-bit words, *merging words*.

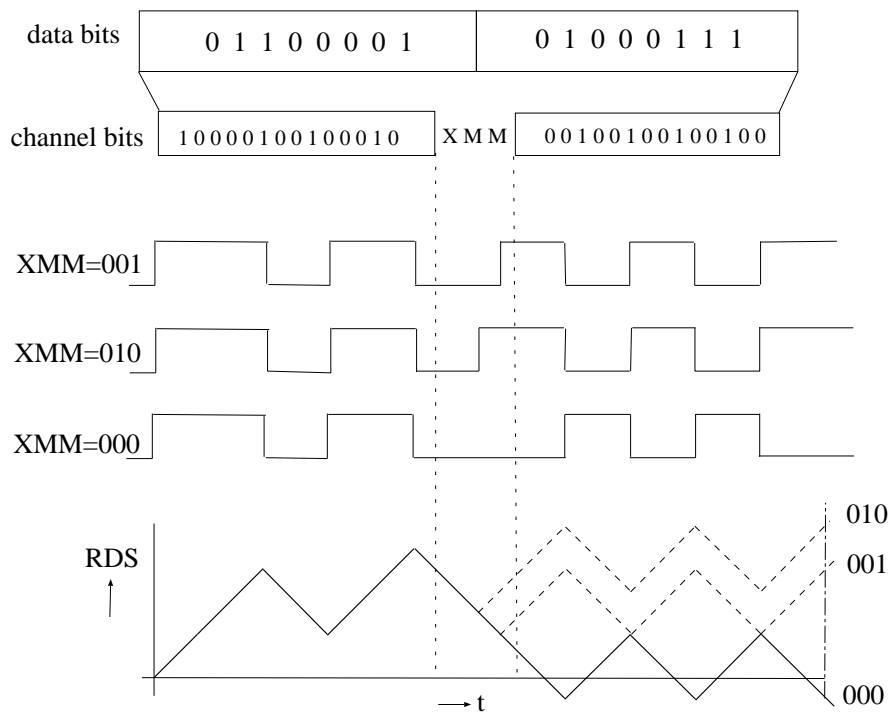


Figure 11.5: Strategy for minimizing the running digital sum (RDS).

There are instances where the merging word, is not uniquely governed by the minimum and maximum runlength requirements. This freedom of choice is utilized for minimizing the power density at the low-frequency end. Figure 11.5 shows the merging process. Eight user bits are translated into 14 channel bits using a look-up table. The 14 bits are cascaded, 'merged', by means of 3-bit merging words in such a way that the runlength conditions continue to be satisfied. For the case shown, the condition that there should be at least two 'zero's between 'one's requires a 'zero' at the first merging bit position. There are thus three alternatives for the merging words, namely '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. '100' in this case.

The codewords and the 3-bit merging words are chosen such that the dk -constraint of the catenation of alternate code words and merging words is satisfied. In the experimental phase of the Compact Disc [278], it was learnt that the suppression of low-frequency components, when only two merging bits are used, is not sufficiently effective. Thus the number of merging bits was increased to three, so providing a greater degree of freedom to set or omit transitions in the merging bits. With three merging bits in 65% of the block catenations a transition can be set or omitted freely.

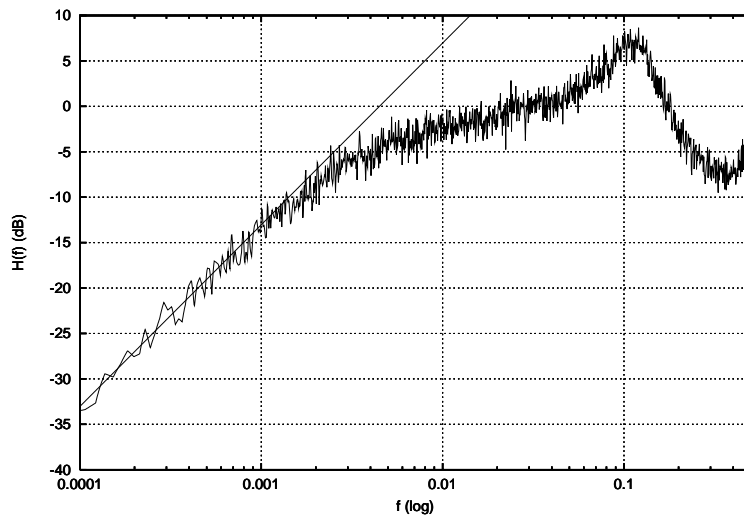


Figure 11.6: Spectrum of classic EFM. The straight line is a least square mean estimate of the low-frequency part of the spectrum.

The more effective low-frequency control is achieved at the expense of $1/17$ of the information rate. Note that the merging process does not guarantee that adequate alternative channel representations can always be found. This means that the RDS of an EFM encoded sequence, N , is unlimited for

judiciously chosen input data. In practice, this did not lead to any (known to the author) difficulties in playing CD's. Apparently, audio and data files are sufficiently random not to become problematic. The Power Spectral Density (PSD) function of classic EFM has been obtained by computer simulation. Results are plotted in Figure 11.6. Noda & Ishizawa [263] published a $d = 1$, rate 12/18 block code, which uses a similar merging rule for dc-suppression and cascading the code words as in EFM. Lin & Liu [222] published a construction technique for $(1, k \geq 4)$ codes. The construction is based on a merging bit rule and the usage of $dklr$ sequences. They developed a rate 8/16, (1,5) dc-free code, which has similar characteristics as the M² code (see Section 11.4.5).

Compatible alternatives to EFM

In principle, better suppression of the low-frequency components can be obtained, without offending the agreed standard for the Compact Disc system, by applying improved merging strategies. For example, by looking more than one symbol ahead, since minimization of the low-frequency content in the short term does not always contribute to longer-term minimization. Improvements of 6-10 dB have been reported [161]. The look-ahead strategy is not used in present equipment.

Immink [159] observed that the standard EFM encoding method is overly restrictive in the choice of the merging words, and that merging words that do not satisfy the dk -constraint can be employed. An alternative encoding scheme, which is fully compatible with standard EFM, was presented. As discussed above, in standard EFM only four 3-bit merging words of the eight possible 3-bit merging words are permitted to be used, namely '001', '010', '000', and '100'. The remaining possible 3-bit merging words, namely '111', '011', '101', and '110' are not used in the standard scheme as they violate the prescribed $d(= 2)$ -constraint. In the alternative scheme [159], the four remaining 3-bit merging words, '111', '011', '101', and '110', are allowed to be used, so that, as a result, all eight 3-bit merging words can be used. Very probably, the above three merging words will be received in error as they violate the $d = 2$ constraint. But, as the 3-bit merging words are skipped anyhow, this will not affect the (user) bit error rate. There are, however, three restrictions: firstly, in the cascade of alternate 14-bit code words and 3-bit merging words the $k(= 10)$ constraint must be satisfied, and secondly in the cascade of alternate 14-bit code words and 3-bit merging words the sync pattern may not be generated, and thirdly, in order to safeguard the 'reliability' of the code words, the number of consecutive 0s at the beginning or end of the two code words joining the merging word should be at least $d(= 2)$. Using the extra merging words gives an extra degree of freedom for suppressing the lf components, and experiments

have shown that a welcome suppression of about 3 dB can be made.

11.5.2 EFMPlus

The Compact Disc (CD) and its extensions CD-ROM and CD-V, introduced in the early 80s, have become a very successful medium for the distribution and storage of audio, MPEG-1 video, and other digital information. Its storage capacity, 680 MByte, is insufficient for graphics-intensive computer applications and high-quality digital video programs.

An extension of the Compact Disc family, the *Digital Versatile Disc (DVD)*, is a new optical recording medium with a storage capacity seven times higher than the conventional Compact Disc. Most of the storage capacity increase is due to improved quality of the light source (red instead of infra-red light) and the objective lens. The storage capacity of the DVD is further enhanced by a complete redesign of the logical format of the disc including a more powerful Reed-Solomon product code (RS-PC) and recording code (EFMPlus). The details of the construction of the rate 8/16, (2,10) EFMPlus code, a sliding-block (d, k) code with suppressed lf-content, will be discussed in the next section.

Design outline

Under EFM rules, see Section 5.6.1, the data bits are translated eight at a time into fourteen channel bits, with runlength parameters $d = 2$ and $k = 10$. In this section we will detail a code with the same runlength constraints as EFM, called *EFMPlus*¹, having a 6% higher rate than classic EFM. EFMPlus has been adopted in the industry standard of the DVD as the channel modulation scheme. The most important design issues of the DVD were that critical parameters such as lf-content and timing should definitely not be compromised. Said parameters are critical as they affect the servos and the timing recovery which are the Achilles' heels of the optical recording system.

EFMPlus is a rate 8/16, sliding-block code with the same runlength parameters as EFM. Dc-control is performed with the surplus words that leave each encoder state. The ACH algorithm is run for a code size, M , that is as large as possible within the complexity constraints. The additional source inputs (> 256) that are made possible in this fashion are employed as alternatives for dc-control (see next section for more details). The complexity of a sliding-block encoder and decoder is essentially governed by the maximum value (weight) of an element of the approximate eigenvector

¹The name EFMPlus is slightly confusing as the acronym EFM stands for Eight to Fourteen Modulation. In EFMPlus there is no such mapping, but the dk constraints are the same as in classic EFM.

(see Section 7.4, page 170). Table 11.8 shows the maximum value (weight) $\max\{v_i\}$ as a function of the code size M . In addition we listed the parameter η , which denotes the relative redundancy, $\eta = 1 - \frac{1}{16} \log_2 M/C(2, 10)$. Note that the maximum code size that can be accommodated for $n = 16$, $d = 2$ and $k = 10$ is 406.

For the given code parameters we note that for code size $M < 352$ the maximum weight is two. A one-round split is sufficient to construct the encoder. We also notice in Table 11.8 that the maximum weight grows very rapidly with mounting code size M . After many trials and considering the diminishing returns, it was decided for a code size $M = 351$. After an initial merging of the $k + 1 = 11$ states we obtain a 3-state FSM. After a single state split, this 3-state FSM can be transformed into a 4-state encoder. Each of the four states of the EFMPlus encoder is characterized by the type of words that enter, or leave, the given state. The states and word sets are characterized by

- Words entering State 1 end with m trailing 'zero's, $m \in \{0, 1\}$;
- Words entering State 2 or 3 end with m trailing 'zero's, $m \in \{2, \dots, 5\}$;
- Words entering State 4 end with m trailing 'zero's, $m \in \{6, \dots, 9\}$.

The words leaving the states are chosen in such a way that the concatenation of words entering a state and those leaving that state obey the ($d = 2, k = 10$) channel constraints. For example, words leaving State 1 start with a runlength of at least two and at most nine 'zero's.

Table 11.8: Code size M , efficiency $\eta = 1 - R/C(2, 10)$ and $\max\{v_i\}$.

M	η	$\max\{v_i\}$
351	0.0246	2
353	0.0237	3
354	0.0232	4
389	0.0075	8
391	0.0067	10
397	0.0041	13
398	0.0037	17
406	0.0000	102

In an analogous manner, we conclude that words leaving State 4 start with at most one 'zero'. Obviously, the sets of words leaving State 1 or 4 have no words in common. Words emerging from State 2 and 3 comply with the above runlength constraints, but they also comply with other conditions.

Words leaving State 2 have been selected such that the first (msb) bit, x_1 , and the thirteenth bit, x_{13} , are both equal 'zero'. Words leaving State 3 have $x_1x_{13} \neq 00$. Any walk through the graph, stepping from state to state, produces a $(d = 2, k = 10)$ -constrained sequence by reading the words tagged to the edges that connect the states. With a computer it can easily be verified that from each of the states at least 351 words are leaving. An encoder is constructed by assigning a source word to each of the 351 edges that leave each state. The encoder requires accommodation for only 256 source words. The excess, 95, words have been used for suppressing the low-frequency power (see next section), the *dc-control*.

The encoder graph is defined in terms of three sets: the inputs, the outputs and the states, and two logical functions: the *output function* and the *next-state function*. The specific codeword, \mathbf{x}_t , transmitted by the encoder at instant t is a function of the source word \mathbf{b}_t that enters the encoder, but depends further on the particular state, s_t , of the encoder. Similarly, the "next" state at instant $t + 1$ is a function of \mathbf{x}_t and s_t . The output function $h(\cdot)$ and the next-state function $g(\cdot)$ can be succinctly written as

$$\mathbf{x}_t = h(s_t, \mathbf{b}_t)$$

$$s_{t+1} = g(s_t, \mathbf{b}_t).$$

Both the output function $h(\cdot)$ and the next-state function $g(\cdot)$ are described by four lists with 351 entries. A part of the output function and the next-state function is listed in Table 11.9. More details can be found in [156].

Table 11.9: Part of the EFMPlus coding table.

i	$h(i, 1), g(i, 1)$	$h(i, 2), g(i, 2)$	$h(i, 3), g(i, 3)$	$h(i, 4), g(i, 4)$
0	001000000001001,1	0100000100100000,2	0010000000001001,1	0100000100100000,2
1	0010000000010010,1	0010000000010010,1	1000000100100000,3	1000000100100000,3
2	0010000100100000,2	0010000100100000,2	1000000000010010,1	1000000000010010,1
3	0010000001001000,2	0100010010000000,4	0010000001001000,2	0100010010000000,4
4	0010000010010000,2	0010000010010000,2	1000000100100000,2	1000000100100000,2
5	0010000000100100,2	0010000000100100,2	1001001000000000,4	1001001000000000,4
6	0010000000100100,3	0010000000100100,3	1000100100000000,4	1000100100000000,4
7	0010000001001000,3	0100000000010010,1	0010000001001000,3	0100000000010010,1
8	0010000010010000,3	0010000010010000,3	1000010010000000,4	1000010010000000,4

Table 11.9 has a column that describes the source (input) word i by an integer between '0' and '255'. The table also shows $h(i, s)$ the 16-bit output to a particular input i when the encoder is in one of the four states s . A 'one' means, as in EFM, a transition pit/land or land/pit, while a 'zero' means the absence of such a transition. The 3rd, 5th, 7th, and 9th columns show the next state function $g(i, s)$. For example, let the encoder graph be

initialized at State 1, and let further the source sequence be '8', '3', '4'. The response to input '8', while being in State 1, equals $h(8, 1) =$ (see Table 11.9) '0010000010010000'. The new state becomes $g(8, 1) = 3$. As a result, the response to input '3', while now being in State 3, is '0010000001001000'. In the next clock cycle, the encoder state becomes $g(3, 3) = 2$. From State 2 with the input equal to '4' we find from the table that the corresponding output is $h(4, 2) =$ '0010000010010000'.

The decoder translates 16-bit words into 8-bit data words. It does not suffice to look at the 16-bit word only, the decoder must also look at symbols at positions 1 and 13 of the upcoming codeword, namely

$$\mathbf{b}_t = h^{-1}(x_t, x_{t+1,1}x_{t+1,13}).$$

Thus decoding of the new code is done by a logic array that translates (16+2) channel bits into 8 bits. In contrast, under EFM rules, it suffices to observe 14 of the 17 channel bits that constitute an EFM codeword.

The encoder defined above can freely accommodate 351 source words. In order to make it possible to use a unique 26-bit sync word, seven candidate words were barred, leaving a code size of 344. As we only need accommodation for 256 source words, the surplus words can be exploited for minimizing the power at low frequencies. The suppression of low-frequency components, or dc-control, is done by controlling the running digital sum (RDS). The 88 surplus words are used as an alternative channel representation of the source words 0,...,87. The full encoder is described by two tables called *main* and *substitute table*, respectively. The source words 0,...,87 can be represented by the designated entries of the main table or alternatively by the entries of the substitute table. For source words 0,...,87 the encoder opts for that particular representation from the main table or the substitute table that minimizes the absolute value of the RDS.

The DVD standard imposes an extra rule for dc-control [318, 157], called *state swapping*. If the encoder is in State 1, the encoder may use the codeword $h(i, 4)$, $0 \leq i \leq 255$, as an alternative for dc-control, provided the run-length constraints are not violated. Similarly, if the encoder is in State 4, it may use the codeword $h(i, 1)$, $0 \leq i \leq 255$, as alternative. In other words, codewords pertaining to States 1 and 4, i.e. $h(i, 1)$ and $h(i, 4)$, both in the main and substitute tables, may be used as alternatives for dc-control, provided the runlengths constraints are strictly obeyed. This so-called state swapping of states 1 and 4 is allowed as decoding can be accomplished unambiguously (only codewords in states 1 and 3 are used for discriminating multiple code words). The state swapping method offers a 2 dB extra reduction of the lf power (see also Table 11.10). Vasic *et al.* presented a modification of the dc-control algorithm of EFMPlus using a look-ahead strategy [329].

11.5.3 Alternatives to EFM schemes

It is of some interest to consider the possibility of redesigning the EFM code, and its variants of various codes rates and to compare the spectral performance.

The EFM code was designed in 1980 before efficient design algorithms, such as ACH and so on, were developed. A second handicap of the EFM design is that at the time of its conception, every gate used for decoding was one too many.² Let us now for academic interest ignore for the time being the complexity issue, and start from scratch. Essentially EFMPlus is a redesign of EFM with a rate 8/16 instead of 8/17. Decoding of EFMPlus requires 1000 instead of the 52 gates of EFM.

An obvious alternative of the rate 8/16, EFMPlus code would have been EFM with two instead of three merging bits. The dc-content of the alternative code, *EFM16a*, (the name we shall use for the code with two instead of three merging bits), can easily be assessed by computer simulation, and the results are shown in Table 11.10. The dc-content can be reduced significantly by a re-assignment of the various words that takes into account the following observation. Observe, for example, that in EFM16a, the 16-bit words '0001000100001000' and '1001000100001000' are alternative channel representations. It can easily be verified that the disparity of both words (after precoding, of course) is zero. This, in fact, means that the encoder has no real option to increase or decrease the RDS with the transmission of those codewords. Obviously, it would be much better if we could redesign the code in such a way that as many source words as possible would have channel representations of zero-disparity. Non-zero-disparity codewords of opposite signs should be paired while zero-disparity codewords may remain single.

It is a straightforward exercise, using Gu & Fuja's method (see Section 5.5.1, page 111), to design a block code according to the above design heuristic. Note that there is no need to rely on Construction 2, page 109. We may construct a block-decodable code with a source size of 260 instead of 257 words. A typical result, note there are many possibilities, called *EFM16b*, offers 8 dB, see Table 11.10, more reduction at the low-frequency end than EFM16a. This is a significant improvement, in particular as the only disadvantage is the extra gate count required for decoding. Note that the EFM16b code requires a full decoding array of 16 bits instead of 14 bits as in EFM16a. An advantage with respect to EFMPlus, which requires a sliding-block decoder of length two, is the absence of error propagation. On

²Note that this requirement was not imposed for the *encoding* hardware, as it was anticipated that there would be a very limited number of master and replication plants. Who, at that time, could envisage that there would be EFM encoders in the households in CD-R and CD-RW players as computer peripheral?

the other side of the balance, we have a 3 dB extra reduction of EFMPlus' lf-content (see Table 11.10).

As $8/15 = 0.5333\dots$, at least in theory, it is possible to construct a rate $8/15$, (2,9) code, and many examples have been given in the literature. A 5-state, rate $8/15$, (2,10) RLL code with decoder window equal to two has been published by Li *et al.* [225]. The code presented by Li *et al.* does not have the virtue of lf suppression. Alternatively, the rate $8/15$, (2,14) *EFM15 code* [157] is an example of a dc-free RLL code. The encoder is a 4-state finite-state machine, where residual codewords are, as in EFMPlus, used as alternatives channel representations to lower the lf-content. An alternative rate $8/15$ construction is possible with, for example, Construction 3, see Section 5.8.2, page 128. In Construction 3 we require, in contrast with Construction 2, where $d = 2$ merging are required, $d - 1 = 1$ merging bit. We could, in principle, deploy the same 14-bit word assignment as in classical EFM. For an example of such a assignment, the reader is referred to the US Patent granted to Tanaka *et al.* [317].

Table 11.10: Performance of EFM-like codes.

<i>Code</i>	$H(10^{-4}f_c)$ (dB)	<i>Sum var.</i>	<i>R</i>	<i>Remark</i>
EFM	-33	16	8/17	Sec. 11.5.1
EFMPlus	-30.5	19	8/16	Sec. 11.5.2
EFMPlus*	-28.5	24	8/16	no state swap
EFM16a	-20	66	8/16	2 merging bits
EFM16b	-28	27	8/16	Sec. 11.5.3
EFM15	-9	220	8/15	-

EFM15, which is very similar to EFMPlus, has a codeword length of 15 bits. The code is a typical sliding-block code, it has four encoder states, and it was constructed using the ACH algorithm after a single round of state splitting. The number of words that can be accommodated depends on the state, and is at most 270. This leaves at most $270 - 256 = 14$ "spare" words that can be used for dc-control. Pairing of the alternative representations has been accomplished in such a way that the words that form a pair differ in a single position, i.e., have unity Hamming distance. This has the advantage that a) the decoding operation is simplified and b) that the alternative representations have an odd or even number of 'one's, which, as was observed experimentally, has a beneficial effect upon the dc-control. Further details, such as coding tables and so on, of EFM15 can be found in the US Patent description [157].

Hayami [124] presented an alternative rate $8/15$, EFM-like (2,10) code. The encoder graph, found with a single round of state splitting, can be

realized by a 7-state FSM. The performance in terms of lf suppression as compared to other EFM-like schemes was not disclosed, and therefore not added to the survey table 11.10. Hayami's code does not use, as EFMPlus and EFM15, a main and a substitute table, for representing specific source words. It uses a state-swapping technique, where, if the dk constraints permit and decoding can be accomplished unambiguously, codewords can be taken from a different state than the current one. This state-swapping procedure can be seen as a generalization of the 1-4 state swapping, which is part of the EFMPlus standard, see Section 11.5.2. Shim & Won presented a rate $8/15$, $(2,12)$ code, which is very similar to EFMPlus [298].

11.5.4 Performance of EFM-like coding schemes

The spectral performance of the various members of the EFM family has been assessed by computer simulation. The outcomes of the simulations have been collected in Table 11.10. The lf-suppression, as presented in Table 11.10, is measured at $10^{-4}f_c$, where the channel bit frequency $f_c = 1$ Hz. If we wish to compare coding schemes of different rate it is standard practice to compare the lf-suppression at, say, 0.0001 times the *user bit* frequency f_b .

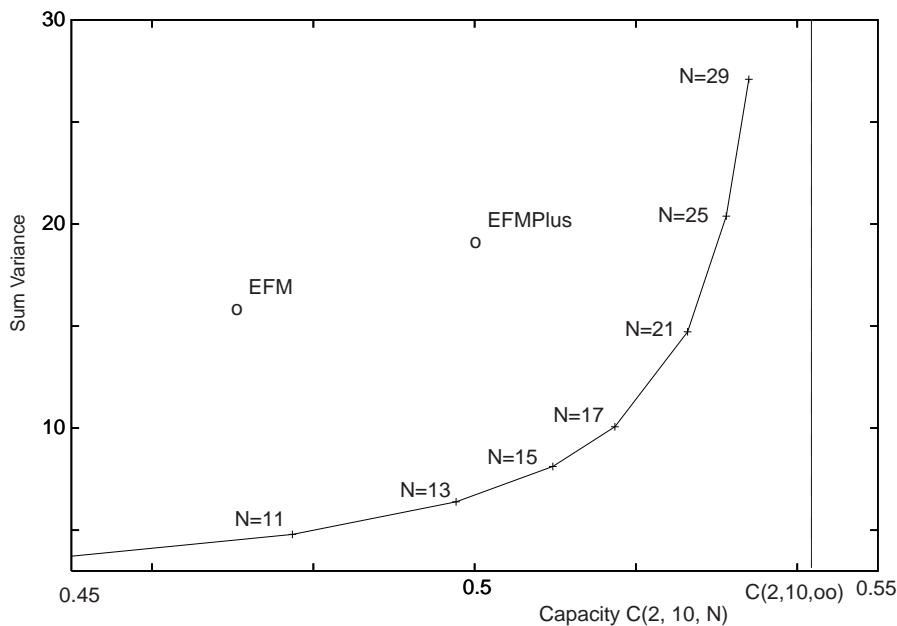


Figure 11.7: Sum variance of maxentropic DCRL sequences with parameters $d = 2$ and $k = 10$ and the digital sum variation N as a parameter. As a comparison we plotted the rate and sum variance of EFM and EFMPlus.

As the frequency $10^{-4}f_c$ Hz is assumed to be in the range of frequencies, where the spectrum $H(f_c)$ has a parabolic shape, the lf-suppression at $10^{-4}f_b$ can be found by multiplying the lf-suppression measured at $10^{-4}f_c$ by R^3 . For example, let $R = 1/2$, then we have to subtract $3 \times 3.1 = 9.3$ dB from the numbers shown in Table 11.10 to obtain the lf-suppression at $10^{-4}f_b$, where $f_b = 1$ Hz. A comparison of the properties of sequences generated under the rules of EFM and EFMPlus with those of maxentropic sequences is shown in Figure 11.7. As we can observe, theory predicts there is quite some room for improvements. For codes of the same rate as EFM and EFMPlus we could, in theory, construct codes that generate sequences with a factor of three smaller sum variance or alternatively a 10 dB extra lf-content suppression. If, on the other hand, we stipulate that the sum variance, and thus the lf-content of EFMPlus is adequate, we may conclude from Figure 11.7 that a code of rate 0.53 ($\approx 8/15$) is possible having the sum variance of EFM. The performance of the rate 8/15, code, EFM15 [157], listed in Table 11.10, is a far cry from the theoretical bound [44].

Other EFM-like codes have been presented by Roth [291]. Braun *et al.* [45] presented coding schemes using long block codes with enumerative coding, as discussed in Chapter 6, that are very close to the predicted maxentropic performance. The typical codeword length in his construction is about 1000 bits, and the hardware required for encoding and decoding is about 5kB.

Chapter 12

Further Reading

There are, to my best knowledge, no textbooks that deal, in a whole, with the theory and practice of coding techniques for the optical and magnetic recording channel, which, by the way, is one of the reasons to write this one. In the past thirty years or so there has grown a staggering literature in this field, but most of the information available is sketchy and is widely dispersed.

There are, however, excellent chapters of books and survey papers in the literature available. Notable is the very early survey paper by Kobayashi in 1971 [208]. Other examples are Chapter 20 by Marcus, Roth & Siegel in *The Handbook of Coding Theory* [236], the survey paper by Immink, Siegel & Wolf [167], which commemorates the 50th anniversary of Shannon's landmark paper, Patel's chapter in [244], and Immink's chapter in [41] provide a state of the art description of coding techniques for the recording channel, respectively. The other chapters in the books mentioned describe the physical context of optical and magnetic recording. Gregory [115] furnishes a comprehensive case study, with emphasis on the signal processing, of the 4:2:2, or D1 format, digital video tape recorder (VTR). The classical work of Cattermole [53] on line coding provides a historical background on dc-balanced and other codes.

The July 1983 issue of the *International Journal of Electronics*, the May 1991 issue of the *IEEE Transactions on Information Theory*, the January 1992 issue of the *IEEE Journal on Selected Areas on Communications*, and the issue of the same Journal published in April 2001, are entirely devoted to coding and signal processing of recording channels. At the end of this chapter, the author assembled a list of books and survey papers that are excellent sources for further study.

The above books and other publications are in the conventional academic setting. An alternative source of information is offered by the patent literature. Note that many constructions of codes originate in the environment of industrial laboratories, and, therefore, it may not be surprising that the

patent literature is an inexhaustible source of information on various coding "methods and apparatuses". The inventions filed are probably far more numerous than anyone suspects who has not made a specialized study of them. According to the European Patent Office, more than 80 percent of man's technical knowledge is described in the patent literature. Patents and other technical articles complement each other. Papers published at conferences will usually become available much earlier than reviewed articles or patents. The review process of archival-quality articles usually takes two years or so, and a patent application is published, "disclosed", 18 months after the filing date of the first-filed counterpart patent application. Patents often describe more complete implementation details and more variations of an idea than do journal articles. The reading of patent descriptions is therefore particularly recommended.

There are some difficulties to overcome when reading the patent literature, which are caused by the technical-legal character of the documents. One of the difficulties is that a patent avoids the usage of every-day terms and uses broader, generic names instead. This practice may give the impression that the patent attorney tries to disguise the subject of the invention. However, the aim of the attorney is to cover as large a technical area as possible. He does not want the scope of the patent to be limited to the device having the every-day name, even though this was the device actually invented, but he extends the patent to the much broader class having the generic name. For example, a whistling kettle is in patent-legal parlance "A device for heating a liquid, preferably water, provided with acoustic signaling for indicating the attainment of the boiling temperature of said liquid".

A second difficulty that has seriously hampered the reading of the patent literature was the difficulty of accessing the patent literature for researchers who are not affiliated with industrial research. This has drastically changed as, recently, many WEB sites were opened, where patent searches can be conducted. For example, the site maintained by the European Patent Office (EPO) with URL address **ep.espacenet.com** offers full (and free) text copies and drawings of patents from essentially all countries that can easily be downloaded. The US Patent Office (USPTO) maintains a very useful web site at **www.uspto.gov**, where free searches and copies of US patents and applications can be obtained.

12.1 Selected literature

H.N. Bertram, *Theory of Magnetic Recording*, Cambridge, UK, Cambridge University Press, 1994.

R.E. Blahut, *Principles and Practice of Information Theory*, Addison Wes-

ley, New York, 1987.

G. Bouwhuis, J. Braat, A. Huijser, J. Pasman, G. van Rosmalen, and K.A.S. Immink, *Principles of Optical Disc Systems*, Adam Hilger Ltd, Bristol and Boston, 1985.

K.W. Cattermole, 'Principles of Digital Line Coding', *Int. J. Electron.*, vol. 55, pp. 3-33, July 1983.

S. Gregory, *Introduction to the 4:2:2 Digital Video Tape Recorder*, Pentech Press, London, 1988.

K.A.S. Immink, 'A Survey of Codes for Optical Disk Recording', *IEEE J. Select. Areas Commun.*, vol.19, no.4, pp.756-764, April 2001.

K.A.S. Immink, P.H. Siegel, and J.K. Wolf, 'Codes for Digital Recorders', *IEEE Trans. Inform. Theory*, vol. 44, pp. 2260-2299, Oct. 1998.

H. Kobayashi, 'A Survey of Coding Schemes for Transmission or Recording of Digital Data', *IEEE Trans. Commun.*, vol. COM-19, pp. 1087-1099, Dec. 1971.

D. Lind and B. Marcus, *Symbolic Dynamics and Coding*, Cambridge University Press, 1995.

B.H. Marcus, P.H. Siegel, and J.K. Wolf, 'Finite-state Modulation Codes for Data Storage', *IEEE Journal on Selected Areas in Communications*, vol. 10, no. 1, pp. 5-37, Jan. 1992

B.H. Marcus, R.M. Roth, and P.H. Siegel, 'Constrained Systems and Coding for Recording Channels', in *Handbook of Coding Theory*, R. Brualdi, C. Huffman, and V. Pless, Eds., Amsterdam, The Netherlands, Elsevier Press, 1996.

C.D. Mee and E.D. Daniel, *Magnetic Recording*, McGraw-Hill Book Company, New York, 1987.

P.H. Siegel, 'Recording Codes for Digital Magnetic Storage', *IEEE Trans. Magn.*, vol. MAG-21, no. 5, pp. 1344-1349, Sept. 1985.

P.H. Siegel and J.K. Wolf, 'Modulation and Coding for Information Storage', *IEEE Commun. Magazine*, vol. 29, no. 12, pp. 68-86, Dec. 1991.

K.C. Pohlman, *The Compact Disc Handbook*, Madison, 1992.

J. Watkinson, *The Art of Digital Audio*, Focal Press, London, 1988.

J. Watkinson, *The Art of Digital Video*, Focal Press, London, 1990.

R.W. Wood, 'Magnetic Recording Systems', *Proceedings IEEE*, vol. 74, pp. 1557-1569, Nov. 1986.

Chapter 13

Author's Biography

Kees A. Schouhamer Immink, obtained M.S. and Ph.D degrees at the Eindhoven University of Technology. He is president and founder of *Turing Machines Inc.*, and since 1995, he is an adjunct professor at the Institute for Experimental Mathematics, Duisburg-Essen University, Germany. In addition, he is affiliated with the National University of Singapore, and the Data Storage Institute, Singapore.

He has contributed to the design and development of a wide variety of consumer-type audio and video recorders such as the LaserVision video disc, Compact Disc, Compact Disc Video, DAT, DV, DCC, DVD, and BluRay Disc system. He holds about 50 issued and pending U.S. patents in various fields.

Dr Immink received wide recognition for his contributions to the technologies of digital video, audio, and data recording. He was inducted into the Consumer Electronics Hall of Fame, elected into the Royal Netherlands Academy of Sciences (KNAW), and named a Knight of the Order of Orange-Nassau. He received an Emmy Award from the US National Television Academy, IEEE Edison Medal, AES Gold and Silver Medals, SMPTE Progress Medal, IEEE Masaru Ibuka Consumer Electronics Award, the Golden Jubilee Award for Technological Innovation awarded by the IEEE IT Society, the IEE Thomson Medal, and the SMPTE Poniatoff Gold Medal. He was awarded fellowships of the IEEE, AES, SMPTE, and IEE.

He served the engineering community as president of the Audio Engineering Society (AES), and as a governor of both the IEEE Consumer Electronics Society and Information Theory Society. He was a member of the IEEE Fellows Committee.

Bibliography

- [1] K.A.S. Abdel-Ghaffar and J.H. Weber, 'Constructing Efficient DC-Free Runlength-Limited Block Codes for Recording Channels', *IEEE Trans. Inform. Theory*, vol. IT-46, no. 4, pp. 1599-1602, July 2000.
- [2] 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, no. 1, pp. 5-22, Jan. 1983.
- [3] R.L. Adler, M. Hassner, and J. Moussouris, 'Method and Apparatus for Generating a Noiseless Sliding Block Code for a (1,7) Channel with Rate 2/3', US Patent 4,413,251, Nov. 1983.
- [4] R.L. Adler, R.K. Brayton, and B.P. Kitchens, 'A Rate 1/3, (5,12) RLL Code', *IBM Techn. Discl. Bul.*, vol. 27, pp. 4722-4724, 1985.
- [5] R.L. Adler, R.K. Brayton, M. Hassner, and B.P. Kitchens, 'A Rate 2/3, (1,6) RLL Code', *IBM Techn. Discl. Bul.*, vol. 27, pp. 4727-4729, 1985.
- [6] N. Alon, E.E. Bergmann, D. Coppersmith, and A.M. Odlyzko, 'Balancing Sets of Vectors', *IEEE Trans. Inform. Theory*, vol. IT-34, no. 1, pp. 128-130, Jan. 1988.
- [7] R.B. Ash, *Information Theory*, Inter-science Publishers, New York, 1965.
- [8] J.J. Ashley, 'Capacity Bounds for Channels with Spectral Nulls', IBM Research Reports RJ 5676, 1987.
- [9] ———, 'A Linear Bound for Sliding Block Decoder Window Size', *IEEE Trans. Inform. Theory*, vol. IT-34, no.3, pp. 389-399, May 1988.
- [10] J.J. Ashley, M. Hilden, P. Perry, and P.H. Siegel, 'Correction to "A Note on the Shannon Capacity of Runlength-Limited Codes"', *IEEE Trans. Inform. Theory*, vol. IT-39, no. 3, pp. 1110-1112, May 1993.

- [11] J.J. Ashley and B.H. Marcus, 'Canonical Encoders for Sliding-block Decoders', *SIAM J. Discrete Math.*, vol. 8, pp. 555-605, 1995.
- [12] ———, 'Two-Dimensional Low-Pass Filtering Codes', *IEEE Trans. Commun.*, vol. 46, no. 6, pp. 724-7, 1998.
- [13] J.J. Ashley and P.H. Siegel, 'A Note on the Shannon Capacity of Run-Length-Limited Codes', *IEEE Trans. Inform. Theory*, vol. IT-33, no. 4, pp. 601-605, July 1987.
- [14] J.J. Ashley, R. Karabed, and P.H. Siegel, 'Complexity and Sliding-block Decodability', *IEEE Trans. Inform. Theory*, vol. IT-42, no. 6, pp. 1925-1947, Nov. 1996.
- [15] ———, 'A Generalized State-Splitting Algorithm', *IEEE Trans. Inform. Theory*, vol. IT-43, no. 4, pp. 1326-1338, July 1997.
- [16] J.J. Ashley, B.H. Marcus, and R.M. Roth, 'Construction of Encoders with Small Decoding Look-ahead for Input-constrained Channels', *IEEE Trans. Inform. Theory*, vol. IT-41, no. 1, pp. 55-76, Jan. 1995.
- [17] J.J. Ashley and B.H. Marcus, 'Time-Varying Encoders for Constrained Systems: An Approach to Limiting Error Propagation', *IEEE Trans. Inform. Theory*, vol. IT-46, no. 3, pp. 1038-1043, May 2000.
- [18] P.M. Aziz, S. Surendran, and P.W. Kemsey, 'Generalized rate $N/(N+1)$ codes', *Electronics Letters*, vol. 35, pp. 710-712, April 1999.
- [19] P.M. Aziz, P.W. Kemsey, and S. Surendran, 'Rate 16/17, (0,5) Modulation Code Apparatus and Method for Partial Response Magnetic Recording Channels, US Patent 6,046,691, April 2000.
- [20] ———, 'Rate 24/25, (0,9) Code Method and System for PRML Recording Channels', US Patent 6,130,629, Oct. 2000.
- [21] P.M. Aziz, M. Pervez, I.M. Hughes, P.W. Kempsey, and S. Surendran, 'General rate $N/(N+1)$, (0, G) Code Construction for Data Coding', US Patent 6,204,781, March 2001.
- [22] J.L.E. Baldwin, 'Method and Apparatus for Processing Digital Signals prior to Recording', US Patent 4,851,837, July 1989.
- [23] S. Al-Bassam and B. Bose, 'On Balanced Codes', *IEEE Trans. Inform. Theory*, vol. IT-36, no. 2, pp. 406-408, March 1990.
- [24] R.L. Beckenhauer and W.J. Schäuble, 'Sync Patterns Encoding System for Run-Length Limited Codes', US Patent 4,146,909, March 1979.

- [25] 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, no. 5, pp. 751-754, Sept. 1983.
- [26] W.R. Bennett, 'Statistics of Regenerative Digital Transmission', *Bell Syst. Tech. J.*, vol. 37, pp. 1501-1542, Nov. 1958.
- [27] M. Berkoff, 'Waveform Compression in NRZI Magnetic Recording', *Proc. IEEE*, vol. 52, pp. 1271-1272, Oct. 1964.
- [28] J. Berstel and D. Perrin, *Theory of Codes*, Academic Press, Orlando, 1985.
- [29] H.N. Bertram, *Theory of Magnetic Recording*, Cambridge, UK, Cambridge University Press, 1994.
- [30] E. Biglieri and G. Caire, 'Power Spectrum of Block-Coded Modulation', *IEEE Trans. Commun.*, vol. COM-42, no. 2/3/4, pp. 1580-1585, Feb/Mar/Apr. 1994.
- [31] G. Bilardi, R. Padovani, and G.L. Pierobon, 'Spectral Analysis of Functions of Markov Chains with Applications', *IEEE Trans. Commun.*, vol. COM-31, no. 7, pp. 853 - 861, July 1983,
- [32] J.A. Bixby and R.A. Ketcham, 'QP, an Improved Code for High Density Digital Recording', *IEEE Trans. Magn.*, vol. MAG-15, pp. 1465-1467, Nov. 1979.
- [33] R.E. Blahut, *Principles and Practice of Information Theory*, Addison Wesley, New York, 1987.
- [34] I.F. Blake, 'The Enumeration of Certain Run Length Sequences', *Information and Control*, vol. 55, pp. 222-237, 1982.
- [35] M. Blaum, P.H. Siegel, G.T. Sincerbox, and A. Vardy, 'Method and Apparatus for Modulation of Multi-Dimensional Data in Holographic Storage', US Patent 5,510,912, April 1996.
- [36] ———, 'Method and Apparatus for Modulation of Multi-Dimensional Data in Holographic Storage', US Patent 5,727,226, March 1998.
- [37] W.G. Bliss, 'Circuitry for Performing Error Correction Calculations on Baseband Encoded Data to Eliminate Error Propagation', *IBM Techn. Discl. Bul.*, vol. 23, pp. 4633-4634, 1981.
- [38] ———, 'An 8/9 Rate Time-Varying Trellis Code for High Density Magnetic Recording', *IEEE Trans. Magn.*, vol. MAG-33, pp. 2746-2748, 1997.

- [39] S.M.C. Borgers, E. de Niet, and P.H.N de With, 'An Experimental Digital VCR with 40mm Drum, Single Actuator and DCT-based Bit-Rate Reduction', *IEEE Trans. Consumer Electr.*, vol. CE-34, pp. 597-605, Aug. 1988.
- [40] B.S. Bosik, 'The Spectral Density of a Coded Digital Signal', *Bell Syst. Tech. J.*, vol. 51, pp. 921-932, April 1972.
- [41] G. Bouwhuis, J. Braat, A. Huijser, J. Pasman, G. van Rosmalen, and K.A.S. Immink, *Principles of Optical Disc Systems*, Adam Hilger Ltd, Bristol and Boston, 1985.
- [42] F.K. Bowers, 'Pulse Code Transmission System', US Patent 2,957,947, Oct. 1960.
- [43] V. Braun, 'On Modulation, Coding and Signal Processing for Optical and Magnetic Recording Systems', Doctoral Thesis, Institute for Experimental Mathematics, University of Essen, Fortschritt-Berichte VDI, Reihe 10, Nr. 504, VDI-Verlag, Dusseldorf, 1997.
- [44] V. Braun and A.J.E.M. Janssen, 'On the Low-frequency Suppression Performance of DC-free Runlength-limited Modulation Codes', *IEEE Trans. Consumer Electr.*, vol. CE-42, no.4, pp. 939-945, Nov. 1996.
- [45] V. Braun and K.A.S. Immink, 'An Enumerative Coding Technique for DC-free Runlength-Limited Sequences', *IEEE Trans on Communications*, vol. 48, pp. 2024-2031, Dec. 2000.
- [46] N.J. Calkin and H.S. Wilf, 'The Number of Independent Sets in a Grid Graph', *SIAM J. Discr. Math.*, vol. 11, pp. 54-60, Feb. 1998.
- [47] J. Campello, B. Marcus, R. New, and B. Wilson, 'Constrained Systems with Unconstrained Positions', *IEEE Trans. Inform. Theory*, vol IT-48, pp. 866-879, 2002.
- [48] G.L. Cariolaro and G.P. Tronca, 'Spectra of Block Coded Digital Signals', *IEEE Trans. Commun.*, vol. COM-22, pp. 1555-1563, Oct. 1974.
- [49] G.L. Cariolaro, G.L. Pierobon, and G.P. Tronca, 'Analysis of Codes and Spectra Calculations', *Int. J. Electron.*, vol. 55, pp. 35-79, no. 1, 1983.
- [50] R.O. Carter, 'Low-disparity Binary Coding System', *Electronics Letters*, vol. 1, pp. 65-68, May 1965.
- [51] ———, 'Low-disparity Binary Coding System', US Patent 3,405,235, Oct. 1968.

- [52] K.W. Cattermole, *Principles of Pulse Code Modulation*, Iliffe Books Ltd, London, 1969.
- [53] ———, 'Principles of Digital Line Coding', *Int. J. Electron.*, vol. 55, pp. 3-33, July 1983.
- [54] K.W. Cattermole and J.J. O'Reilly, *Problems of Randomness in Communication Engineering*, vol. 2, Pentech Press, London, 1984.
- [55] J.K. Cavers and R.F. Marchetto, 'A New Coding Technique for Spectral Shaping of Data', *IEEE Trans. Commun.*, vol. COM-40, no. 9, pp. 1418-1422, 1992.
- [56] P. Chaichanavong and B. Marcus, 'Optimal Block-Type-Decodable Encoders for Constrained Systems', *IEEE Trans. Inform. Theory*, vol. IT-49, no. 5, pp. 1231-1250, May 2003.
- [57] R. Cideciyan, F. Dolivo, R. Hermann, W. Hirt, and W. Schott, 'A PRML System for Digital Magnetic Recording', *IEEE Journal Selected Areas in Communications*, vol. 10, pp. 38-56, Jan. 1992.
- [58] R. Cideciyan, E. Eleftheriou, B.H. Marcus, D.S. Modha, 'Maximum Transition Run Codes for Generalized Partial Response Channels', *IEEE Journal Selected Areas in Communications*, vol. 19, pp. 619-634, April 2001.
- [59] T.M. Chien, 'Upper Bound on the Efficiency of Dc-constrained Codes', *Bell Syst. Tech. J.*, vol. 49, pp. 2267-2287, Nov. 1970.
- [60] W. Coene, 'Combi-codes for Dc-free Runlength-Limited Coding', *IEEE Trans. Consumer Electr.*, vol. CE-46, no.4, pp. 1082-1087, Nov. 2000.
- [61] M. Cohn, G.V. Jacoby, and C.A. Bates III, 'Data Encoding Method and System Employing Two-thirds Code Rate with Full Word Look-ahead', US Patent 4,337,458, June 1982.
- [62] 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.
- [63] J.D. Coker, D.T. Flynn, R.L. Galbraith, and T.C. Truax, 'Method and Apparatus for Implementing a Set Rate Code for Data Channels With Alternate 9-bit Codewords and 8-bit Codewords', US Patent 5,784,010, July 1998.

- [64] G. Copeland and B. Tezcan, 'Disparity and Transition Density Control System and Method', US Patent 6,304,196, Oct. 2001.
- [65] D. Coppersmith and B.P Kitchens, 'Run-length Limited Code without DC Level', US Patent 4,675,650, June 1987.
- [66] T.M. Cover, 'Enumerative Source Coding', *IEEE Trans. Inform. Theory*, vol. IT-19, no. 1, pp. 73-77, Jan. 1973.
- [67] S. Datta and S.W. McLaughlin, 'An Enumerative Method for Runlength-Limited Codes: Permutation Codes', *IEEE Trans. Inform. Theory*, vol. IT-45, no. 6, pp. 2199-2204, Sept. 1999.
- [68] ———, 'Optimal Codes for M -ary Runlength-Constrained Channels', *IEEE Trans. Inform. Theory*, vol. IT-47, no. 5, pp. 2069-2078, July 2001.
- [69] N.R. Davie, M.A. Hassner, T.D. Howell, R. Karabed, and P.H. Siegel, 'Method and Apparatus for Asymmetrical RLL Coding', US patent 4,949,196, Aug. 1990.
- [70] R.H. Deng and M.A. Herro, 'DC-Free Coset Codes', *IEEE Trans. Inform. Theory*, vol. IT-34, no. 4, pp. 786-792, July 1988.
- [71] A.J.M. Denissen and L.M.G.M. Tolhuizen, 'Apparatus and Methods for Use in Transmitting and Receiving a Digital Signal, and a Record Carrier Obtained by a Transmission Apparatus or Method', US Patent 5,671,236, Sept. 1997.
- [72] J.S. Eggenberger and P. Hodges, 'Sequential Encoding and Decoding of Variable Word Length, Fixed Rate Data Codes', US Patent 4,115,768, 1978.
- [73] J.S. Eggenberger and A.M. Patel, 'Method and Apparatus for Implementing Optimum PRML Codes', US Patent 4,707,681, Nov. 1987.
- [74] E. Eleftheriou and R.D. Cideciyan, 'On Codes Satisfying M th-Order Running Digital Sum Constraints', *IEEE Trans. Inform. Theory*, vol. IT-37, no. 5, pp. 1294-1313, Sept. 1991.
- [75] T. Etzion, 'Cascading Methods for Runlength-Limited Arrays', *IEEE Trans. Inform. Theory*, vol. IT-43, no. 1, pp. 319-324, Jan. 1997.
- [76] I.J. Fair, W.D. Gover, W.A. Krzymien, and R.I. MacDonald, 'Guided Scrambling: A New Line Coding Technique for High Bit Rate Fiber Optic Transmission Systems', *IEEE Trans. Commun.*, vol. COM-39, no. 2, pp. 289-297, Feb. 1991.

- [77] I.J. Fair, Q. Wang, and V.K. Bhargava, 'Polynomials for Guided Scrambling Line Codes' *IEEE Journal on Selected Areas in Communications*, vol.13,, pp. 499-509, April 1995.
- [78] ———, 'Characteristics of Guided Scrambling Encoders and Their Coded Sequences', *IEEE Trans. Inform. Theory*, vol. IT-43, pp. 342-347, Jan. 1997.
- [79] J.L. Fan and A.R. Calderbank, 'A Modified Concatenated Coding Scheme with Applications to Magnetic Recording', *IEEE Trans. Inform. Theory*, vol. IT-44, pp. 1565-1574, July 1998.
- [80] J.L. Fan, B.H. Marcus, and R.M. Roth, 'Lossless Sliding Block Compression of Constrained Systems', *IEEE Trans. Inform. Theory*, vol. IT-46, pp. 624-633, March 2000.
- [81] W. Feller, *An Introduction to Probability Theory and Its Applications, Volume I*, Wiley and Sons Inc., New York, 1959.
- [82] K.D. Fisher and J. Fitzpatrick, 'Rate 24/25 Modulation Code for PRML Recording Channels', US Patent 5,757,294, May 1998.
- [83] B. Fitinghof and M. Mansuripur, 'Method and Apparatus for Implementing Post-Modulation Error Correction Coding Scheme, US Patent 5,311,521, May 1994.
- [84] J. Fitzpatrick and K.J. Knudson, 'Rate 16/17, ($d = 0, G = 6/I = 7$) Modulation Code for a Magnetic Recording Channel', US Patent 5,635,933, June 1997.
- [85] S. Forchhammer and J. Justesen, 'Bounds on the Capacity of Constrained Two-dimensional Codes', *IEEE Trans. Inform. Theory*, vol. IT-46, pp. 2659-2666, Nov. 2000.
- [86] K. Forsberg and I.F. Blake, 'The Enumeration of (d, k) Sequences', Proc. 26th Allerton Conf. on Communications, Control, and Computing, Montecelli, pp. 471-472, Sept. 1988.
- [87] P.A. Franaszek, 'Coding for Constrained Channels: A Comparison of Two Approaches', *IBM J. Res. Develop.*, vol. 33, no. 6, 602-608, Nov. 1989.
- [88] ———, 'Apparatus for Encoding Unconstrained Data onto a (1,7) Format with Rate 2/3, US Patent 4,488,142, Dec. 1984.
- [89] ———, 'Sequence-State Methods for Run-length-limited Coding', *IBM J. Res. Develop.*, vol. 14, pp. 376-383, July 1970.

- [90] ———, 'Run-length-limited Variable Length Coding with Error Propagation Limitation', US Patent 3,689,899, Sept. 1972.
- [91] ———, 'On Future-dependent Block Coding for Input-restricted Channels', *IBM J. Res. Develop.*, vol. 23, pp. 75-81, 1979.
- [92] ———, 'Synchronous Bounded Delay Coding for Input-restricted Channels', *IBM J. Res. Develop.*, vol. 24, 43-48, 1980.
- [93] ———, 'A General Method for Channel Coding', *IBM J. Res. Develop.*, vol. 24, pp. 638-641, 1980.
- [94] ———, 'Construction of Bounded Delay Codes for Discrete Noiseless Channels', *IBM J. Res. Develop.*, vol. 26, no. 4, pp. 506-514, July 1982.
- [95] ———, 'On Synchronous Variable-Length Coding for Discrete Noiseless Channels', *Information and Control*, vol. 15, pp. 155-164, 1969.
- [96] ———, 'Sequence-State Encoding for Digital Transmission', *Bell Syst. Tech. J.*, vol. 47, pp. 143-157, Jan. 1968.
- [97] J.N. Franklin and J.R. Pierce, 'Spectra and Efficiency of Binary Codes without DC', *IEEE Trans. Commun.*, vol. COM-20, pp. 1182-1184, Dec. 1972.
- [98] L.E. Franks, *Signal Theory*, Prentice-Hall, Inc., Englewood Cliffs, New Jersey, 1969.
- [99] L.J. Fredrickson, 'A $(d, k, c) = (0, 3, 5/2)$ Rate 8/10 Modulation Code', *IEEE Trans. Magn.*, vol. MAG-26, no. 5, pp. 2318-2320, Sept. 1990.
- [100] ———, 'On the Shannon Capacity of DC- and Nyquist-Free Codes', *IEEE Trans. Inform. Theory*, vol. IT-37, no. 3, pp. 918-923, May 1991.
- [101] C.V. Freiman and A.D. Wyner, 'Optimum Block Codes for Noiseless Input Restricted Channels', *Information and Control*, vol. 7, pp. 398-415, 1964.
- [102] G. Freiman and S. Litsyn, 'Asymptotically Exact Bounds on the Size of High-Order Spectral-Null Codes', *IEEE Trans. Inform. Theory*, vol. IT-45, no. 6, pp. 1798-1807, Sept. 1999.
- [103] C.A. French and J.K. Wolf, 'Alternative Modulation Codes for the Compact Disc', *IEEE Trans. Consumer Electronics*, vol. CE-34, no. 4, pp. 908-913, Nov. 1988.

- [104] S. Fukuda, Y. Kojima, Y. Shimpuku, and K. Odaka, '8/10 Modulation Codes for Digital Magnetic Recording', *IEEE Trans. Magn.*, vol. MAG-22, no. 5, pp. 1194-1196, Sept. 1986.
- [105] P. Funk, 'Run-length-limited Codes with Multiple Spacing', *IEEE Trans. Magn.*, vol. MAG-18, no. 2, pp. 772-775, March 1982.
- [106] A. Gabor, 'Adaptive Coding for Self-Clocking Recording', *IEEE Trans. Electronic Computers*, vol. EC-16, pp. 866-868, Dec. 1967.
- [107] P. Galko and S. Pasupathy, 'The Mean Power Spectral Density of Markov Chain Driven Signals', *IEEE Trans. Inform. Theory*, vol. IT-27, no. 6, pp. 746-754, Nov. 1981.
- [108] R.G. Gallager, *Information Theory and Communication*, New York, Wiley, 1968.
- [109] F.R. Gantmacher, *Matrix Theory*, New York: Chelsea, 1960.
- [110] 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.
- [111] E.N. Gilbert, 'Synchronization of Binary Messages', *IEEE Trans. Inform. Theory*, vol. IT-6, no. 5, pp. 470-477, Sept. 1960.
- [112] S.W. Golomb, *Shift Register Sequences*, Holden-Day, Inc., San Francisco, 1967.
- [113] E. Gorog, 'Redundant Alphabets with Desirable Frequency Spectrum Properties', *IBM J. Res. Develop.*, vol. 12, pp. 234-241, May 1968.
- [114] M.D. Gray, 'Variable Rate Bit Inserter for Digital Data Storage', US Patent 5,815,514, Sept. 1998.
- [115] S. Gregory, *Introduction to the 4:2:2 Digital Video Tape Recorder*, Pentech Press, London, 1988.
- [116] L.J. Greenstein, 'Spectrum of a Binary Signal Block Coded for DC Suppression', *Bell Syst. Tech. J.*, vol. 53, pp. 1103-1126, July 1974.
- [117] J.M. Griffiths, 'Binary Code Suitable for Line Transmission', *Electronics Letters*, vol. 5, pp. 79-81, 1969.
- [118] —, 'Low Disparity Binary Codes', US Patent 3,631,471, Dec. 1971.
- [119] J. Gu and T. Fuja, 'A New Approach to Constructing Optimal Block Codes for Runlength-Limited Channels', *IEEE Trans. Inform. Theory*, vol. IT-40, no. 3, pp. 774-785, 1994.

- [120] L.J. Guibas and A.M. Odlyzko, 'Maximal Prefix-Synchronized Codes', *SIAM J. Applied Math.*, vol. 35, no. 2, pp. 401-418, 1978.
- [121] N.H. Hansen, 'A Head-Positioning System using Buried Servos', *IEEE Trans. Magn.*, vol. MAG-17, no. 6, pp. 2735-2738, Nov. 1981.
- [122] G.H. Hardy and E.M. Wright, *An Introduction to the Theory of Numbers*, 5th Ed., Oxford, England: Clarendon Press, 1979.
- [123] M.A. Hassner, N. Heise, W. Hirt, B.M. Trager, 'Method and Means for Invertibly Mapping Binary Sequences into Rate 2/3, (1, k) Run-Length-Limited Coded Sequences with Maximum Transition Density Constraints', US Patent 6,195,025, Feb. 2001.
- [124] A. Hayami, 'Eight-to-fifteen Modulation using no Merging bit and Optical Recording or Reading Systems Based Thereon', US Patent 6,297,753, Oct. 2001.
- [125] M.K. Haynes, 'Magnetic Recording Techniques for Buried Servos', *IEEE Trans. Magn.*, vol. MAG-17, no. 6, pp. 2730-2734, Nov. 1981.
- [126] J.F. Heanue, M.C. Bashaw, and L. Hesselink, 'Volume Holographic Storage and Retrieval of Digital Data', *Science*, pp. 749-752, 1994.
- [127] ———, 'Channel Codes for Digital Holographic Data Storage', *J. Opt. Soc. Am.*, vol. 12, 1995.
- [128] M. Hecht and A. Guida, 'Delay Modulation', *Proceedings IEEE*, vol. 57, pp. 1314-1316, July 1969.
- [129] C.D. Heegard, B.H. Marcus, and P.H. Siegel, 'Variable-length State Splitting with Applications to Average Runlength-constrained (ARC) Codes', *IEEE Trans. Inform. Theory*, vol. IT-37, no. 3, pp. 759-777, May 1991.
- [130] 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.
- [131] P.S. Henry, 'Zero Disparity Coding System', US Patent 4,309,694, Jan. 1982.
- [132] W. Hirt, M. Hassner, and N. Heise, 'IrDA-VFlr (16 Mb/s): Modulation Code and System Design', *IEEE Personal Communication*, pp. 58-71, Feb. 2001.

- [133] T. Himeno, M. Tanaka, T. Katoku, K. Matsumoto, M. Tamura, and H. Min-Jae, 'High-density Magnetic Tape Recording by a Nontracking Method', *Electronics and Communications in Japan, Part 2*, Vol. 76, no. 5, pp. 83-93, 1993
- [134] J. Hogan, R.M. Roth, and G. Ruckenstein, 'Nested Input-Constrained Codes', *IEEE Trans. Inform. Theory*, vol. IT-46, pp. 1302-1316, July 2000.
- [135] K. Hole and Ø. Ytrehus, 'Improved Coding Techniques for Partial-Response Channels', *IEEE Trans. Inform. Theory*, vol. IT-40, no. 2, pp.482-493, March 1994.
- [136] H.D.L. Hollmann, 'A Block-Decodable $(d, k) = (1, 8)$ Runlength-Limited rate 8/12 code', *IEEE Trans. Inform. Theory*, vol. IT-40, no. 4, pp. 1292-1295, July 1994.
- [137] ———, 'On the Construction of Bounded-delay Encodable Codes for Constrained Systems', *IEEE Trans. Inform. Theory*, vol. IT-41, no. 5, pp. 1354-1378, Sept. 1995.
- [138] ———, 'Bounded-delay-encodable block-decodable codes for constrained systems', *IEEE Trans. Inform. Theory*, vol. IT-42, no. 6, pp. 1957-1970, Nov. 1996.
- [139] ———, *Modulation Codes*, Thesis Eindhoven University of Technology, Dec. 1996.
- [140] ———, 'On an Approximate Eigenvector Associated with a Modulation Code', *IEEE Trans. Inform. Theory*, vol. IT-43, pp. 1672-1678, Sept. 1997.
- [141] H.D.L. Hollmann and K.A.S. Immink, 'Performance of Efficient Balanced Codes', *IEEE Trans. Inform. Theory*, vol. IT-37, no. 3, pp. 913-918, May 1991.
- [142] A. Hoogendoorn, 'Digital Compact Cassette', *Proceedings IEEE*, vol. 82, no. 10, pp. 1479-1489, Oct. 1994.
- [143] T. Horiguchi and K. Morita, 'An Optimization of Modulation Codes in Digital Recording', *IEEE Trans. Magn.*, vol. MAG-12, no. 6, pp. 740-742, Nov. 1976.
- [144] D.G. Howe and H.M. Hilden, 'Shift Error Propagation in (2,7) Modulation Code', *IEEE Journal on Selected Areas in Communications*, vol. 10, no. 1, pp. 223-232, Jan. 1992.

- [145] 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.
- [146] ———, 'Statistical Properties of Selected Recording Codes', *IBM J. Res. Develop.*, vol. 33, no. 1, pp. 60-73, Jan. 1989.
- [147] K.A.S. Immink, 'Construction of Binary DC-constrained Codes', *Philips J. Res.*, vol. 40, pp. 22-39, 1985.
- [148] ———, 'Spectral Null Codes', *IEEE Trans. Magn.*, vol. MAG-26, no. 2, pp. 1130-1135, March 1990.
- [149] ———, 'Block-decodable Runlength-limited Codes Via Look-ahead Technique', *Philips J. Res.*, vol. 46, no. 6, pp. 293-310, 1991.
- [150] ———, *Coding Techniques for Digital Recorders*, Prentice-Hall International (UK) Ltd., Englewood Cliffs, New Jersey, 1991.
- [151] ———, 'The Digital Versatile Disc (DVD): System Requirements and Channel Coding', *SMPTE Journal*, vol. 105, pp. 483-489, no. 8, Aug. 1996.
- [152] ———, 'A rate 4/6, ($d = 1, k = 11$) Block-decodable Runlength-limited Code', *IEEE Trans. Inform. Theory*, vol. IT-42, no. 5, pp. 1551-1553, Sept. 1996.
- [153] ———, 'Methods and Devices for Converting a Sequence of m -bit Information Words to a Modulated Signal and Including that Signal on a Record Carrier, Devices for Decoding that Signal and Reading it from a Record Carrier, and that Signal', US Patent 5,642,113, June 1997.
- [154] ———, 'A Practical Method for Approaching the Channel Capacity of Constrained Channels', *IEEE Trans. Inform. Theory*, vol. IT-43, no. 5, pp. 1389-1399, Sept. 1997.
- [155] ———, 'Weakly Constrained Codes', *Electronics Letters*, vol. 33, no. 23, pp. 1943-1944, Nov. 1997.
- [156] ———, 'Method of Converting a Series of m -bit Information Words to a Modulated Signal, Method of Producing a Record Carrier, Coding Device, Device, Decoding Device, Recording Device, Reading Device, Signal as well as Record Carrier', US Patent 5,696,505, Dec. 1997.
- [157] ———, 'Method of Converting a Series of m -bit Information Words to a Modulated Signal, Method of Producing a Record Carrier, Coding Device, Device, Recording Device, Signal, as well as a Record Carrier', US Patent 5,790,056, Aug. 1998.

- [158] ———, 'A Survey of Codes for Optical Disk Recording', *IEEE J. Select. Areas Commun.*, vol.19, no.4, pp.756–764, April 2001.
- [159] ———, 'Method of Converting a Series of m -bit Information Words into a Modulated Signal, US Patent 6,768,432, July 2004.
- [160] K.A.S. Immink and G.F.M. Beenker, 'Binary Transmission Codes with Higher Order Spectral Zeros at Zero Frequency', *IEEE Trans. Inform. Theory*, vol. IT-33, no. 3, pp. 452-454, May 1987.
- [161] K.A.S. Immink and U. Gross, 'Optimization of Low-frequency Properties of Eight-to-Fourteen Modulation', *The Radio and Electronic Engineer*, vol. 53, no. 2, pp. 63-66, Feb. 1983.
- [162] K.A.S. Immink and H.D.L. Hollmann, 'Prefix-synchronized Run-length-limited Sequences', *IEEE Journal on Selected Areas in Communications.*, vol. 10, no. 1, pp. 214-222, Jan. 1992.
- [163] K.A.S. Immink and A.J.E.M. Janssen, 'Error Propagation Assessment of Enumerative Coding Schemes', *IEEE Trans. Inform. Theory*, vol. IT-45, no. 7, Nov. 1999.
- [164] K.A.S. Immink, J.Y. Kim, S.W. Suh, and S.K. Ahn, 'Extremely Efficient Dc-free RLL codes for Optical Recording', *IEEE Trans. Commun.*, vol COM-51, no. 3, pp. 326-331, March 2003.
- [165] K.A.S. Immink and H. Ogawa, 'Method for Encoding Binary Data', US Patent 4,501,000, Feb. 1985.
- [166] K.A.S. Immink and L. Patrovics, 'Performance Assessment of DC-Free Multimode Codes', *IEEE Trans. Commun.*, vol. 45, no. 3, March 1997.
- [167] K.A.S. Immink, P.H. Siegel, and J.K. Wolf, 'Codes for Digital Recorders', *IEEE Trans. Inform. Theory*, vol. IT-44, pp. 2260-2299, Oct. 1998.
- [168] K.A.S. Immink and A.J. van Wijngaarden, 'Simple High-rate Constrained Codes', *Electronics Letters*, vol. 32, no. 20, pp. 1877, Sept 1996
- [169] H. Ino, T. Sato, and T. Nakagawa, 'Modulating Method, Modulating Device and Demodulating Device', US Patent 5,506,581, April 1996.
- [170] J. Isailovic, *Videodisc and Optical Memory Systems*, Prentice-Hall, Inc., Englewood Cliffs, New Jersey, 1985.

- [171] M. Isozaki, 'Digital Recording System with (8,16,2,5) Run Length Limited (RLL) Code', US Patent 5,198,813, Mar. 1993.
- [172] G.V. Jacoby, 'A New Look-Ahead Code for Increasing Data Density', *IEEE Trans. Magn.*, vol. MAG-13, no. 5, pp. 1202-1204, Sept. 1977.
- [173] ———, 'Method and Apparatus for Encoding and Recovering Binary Digital Data', US Patent 4,323,931, Apr. 1982.
- [174] G.V. Jacoby and R. Kost, 'Binary Two-thirds Rate Code with Full Word Look-Ahead', *IEEE Trans. Magn.*, vol. MAG-20, no. 5, pp. 709-714, Sept. 1984.
- [175] A.J.E.M. Janssen and K.A.S. Immink, 'An Entropy Theorem for Computing the Capacity of Weakly (d, k) -constrained Sequences', *IEEE Trans. Inform. Theory*, vol. IT-46, pp. 1034-1038, May 2000.
- [176] F. Jorgensen, *The Complete Handbook of Magnetic Recording*, TAB Books, Blue Ridge Summit, Pennsylvania, July 1980.
- [177] J. Justesen, 'Information Rates and Power Spectra of Digital Codes', *IEEE Trans. Inform. Theory*, vol. IT-28, no. 3, pp. 457-472, May 1982.
- [178] ———, 'Calculation of Power Spectra for Block Coded Signals', *IEEE Trans. Commun.*, vol. COM-49, no. 3, pp. 389 - 392, Mar. 2001.
- [179] J. Justesen and T. Hoholdt, 'Maxentropic Markov Chains', *IEEE Trans. Inform. Theory*, vol. IT-30, no. 4, pp. 665-667, July 1984.
- [180] J.A.H. Kahlman and K.A.S. Immink, 'Channel Code with Embedded Pilot Tracking Tones for DVCR', *IEEE Trans. Consumer Electr.*, vol. CE-41, no. 1, pp. 180-185, Feb. 1995.
- [181] ———, 'Device for Encoding/Decoding N -bit Source Words into Corresponding M -bit Channel Words, and Vice Versa', US Patent 5,477,222, Dec. 1995.
- [182] J.A.H. Kahlman, K.A.S. Immink, G. van den Enden, T. Nakagawa, Y. Shimpuku, T. Narahara, and K. Nakamura, 'Modulation Apparatus/Method, Demodulation Apparatus/Method and Program Presenting Medium', US Patent 6,677,866, Jan. 2004.
- [183] H. Kamabe, 'Minimum Scope for Sliding Block Decoder Mappings', *IEEE Trans. Inform. Theory*, vol. IT-35, no. 6, pp. 1135-1340, Nov. 1989.

- [184] ———, 'Spectral Lines of Codes as Functions of Finite Markov Chains', *IEEE Trans. Inform. Theory*, vol. IT-37, no. 3, pp. 927-941, May 1991.
- [185] K. Kanota and M. Nagai, 'Signal Processing Apparatus Selecting a Scrambled Signal having a Desired DC component from among a Plurality of Scrambled Signal Obtained by Scrambling an Input-data Signal with Respective Pseudo-random Signals', US Patent 5,122,912, June 1992.
- [186] R. Karabed and B.H. Marcus, 'Sliding-Block Coding for Input-Restricted Channels', *IEEE Trans. Inform. Theory*, vol. IT-34, no. 1, pp. 2-26, Jan. 1988.
- [187] R. Karabed and P.H. Siegel, 'Matched Spectral-Null Codes for Partial-Response Channels', *IEEE Trans. Inform. Theory*, vol. IT-37, no. 3, pp. 818-855, May 1991.
- [188] ———, 'A 100% Efficient Sliding-block Code for the Charge-constrained Runlength Limited Channel with Parameters $(d, k, c) = (1, 3, 3)$ ', IEEE International Symposium Inform. Theory, Budapest, June 1992.
- [189] ———, 'Even-mark-modulation for Optical Recording', US Patent 4,870,414, Sept. 1989.
- [190] ———, 'Even Mark Modulation for Optical Recording', Proceedings 1989 IEEE Int. Conf. on Communications, vol. 3, Session 53.5.1, pp. 1628-1632, June 1989.
- [191] R. Karabed, P.H. Siegel, and E. Soljanin, 'Constrained Coding for Binary Channels with high Intersymbol Interference', *IEEE Trans. Inform. Theory*, vol. IT-45, no. 6, pp. 1777-1797, Sept. 1999.
- [192] N. Kashyap and P.H. Siegel, 'Sliding-Block Decodable Encoders Between (d, k) Runlength-Limited Constraints of Equal Capacity', *IEEE Trans. Inform. Theory*, vol. IT-50, no. 6, pp. 1327-1331, June 2004.
- [193] Y. Katayama, T. Nishiya, H. Yamakawa, T. Katou, and S. Taira, 'Asymmetrical DC control Code for Dual-layered Optical Disks', *Journal of the Magnetism Society of Japan*, vol. 25, no. 3, pt. 2, pp. 457-458, 2001.
- [194] Y. Katayama, T. Katou, Nishiya, H. Yamakawa, and S. Taira, 'Rate 8/14 Asymmetrical DC Control Code for Optical Disks', Fourth IEEE Pacific Rim Conference on Lasers and Electro-Optics, CLEO/Pacific Rim 2001, pages II-8 - II-9 vol.2, July 2001.

- [195] A. Kato and K. Zeger, 'On the Capacity of Two-Dimensional Run-Length Constrained Channels', *IEEE Trans. Inform. Theory*, vol. IT-45, no. 5, pp. 1527-1540, July 1999.
- [196] W.H. Kautz, 'Fibonacci Codes for Synchronization Control', *IEEE Trans. Inform. Theory*, vol. IT-11, pp. 284-292, 1965.
- [197] K. Kayanuma, C. Noda, and T. Iwanaga, 'Eight to Twelve Modulation Code for High Density Optical Disk', Proc. ISOM03 Conference, We-F-45, Nov. 2003.
- [198] L. Ke and M.W. Marcellin, 'A New Construction for n -Track (d, k) Codes with Redundancy', *IEEE Trans. Inform. Theory*, vol. IT-41, no. 4, pp. 1107-1115, July 1995.
- [199] J.G. Kemeny and J.L. Snell, *Finite Markov Chains*, Van Nostrand, London, 1960.
- [200] K.J. Kerpez, 'The Power Spectral Density of Maximum Entropy Charge Constrained Sequences', *IEEE Trans. Inform. Theory*, vol. IT-35, no. 3, pp. 692-695, May 1989.
- [201] ———, 'Runlength Codes from Source Codes', *IEEE Trans. Inform. Theory*, vol. IT-37, no. 3, pp. 682-687, May 1991.
- [202] K.J. Kerpez, A. Gallopoulos, and C. Heegard, 'Maximum Entropy Charge-Constrained Run-Length Codes', *IEEE Journal on Selected Areas in Communications*, vol. 10, no. 1, pp. 242-253, Jan. 1992.
- [203] G. Khachatryan and K.A.S. Immink, 'Construction of Simple Runlength-Limited Codes', *Electronics Letters*, vol. 35, no 2, pp. 140, 1999.
- [204] M.J. Kim, '7/13 Channel Coding and Decoding Method Using RLL(2,25) Code', US Patent 6,188,336, Feb. 2001.
- [205] Y.J. Kim, B.M. Jin, and K.R. Cho, 'Coding/decoding System of Bit Insertion/Manipulation Line Code for High-Speed Optical Transmission System', US Patent 6,333,704, Dec 2001.
- [206] D.E. Knuth, 'Efficient Balanced Codes', *IEEE Trans. Inform. Theory*, vol. IT-32, no. 1, pp. 51-53, Jan. 1986.
- [207] H. Kobayashi and D.T. Tang, 'Application of Partial Response Channel Coding to Magnetic Recording Systems', *IBM J. Res. Develop.*, vol. 14, pp. 368-375, July 1970.

- [208] H. Kobayashi, 'A Survey of Coding Schemes for Transmission or Recording of Digital Data', *IEEE Trans. Commun.*, vol. COM-19, pp. 1087-1099, Dec. 1971.
- [209] R. Kuki and K. Saeki, 'Encoder/decoder System with Suppressed Error Propagation', US Patent 6,097,320, Aug. 2000.
- [210] A. Kunisa, 'Control of Parameters of Channel Constraints Using Guided Scrambling for Digital Recording', PhD Thesis, Ehime University, Aug. 1999.
- [211] ———, 'Runlength Control Based on Guided Scrambling for Digital Magnetic Recording', *IEICE Trans. on Electronics*, vol. E82-C, pp. 2209-2217, No. 12, Dec. 1999.
- [212] ———, 'Runlength Violation of Weakly Constrained Code', *IEEE Trans. Commun.*, vol. COM-50, pp. 1-6, Jan. 2002.
- [213] ———, 'Symbol Error Probability for Guided Scrambling Over a Recording Channel', *IEEE Trans. Inform. Theory*, vol. IT-50, no. 2, pp. 344-349, Feb. 2004.
- [214] A. Kunisa, S. Takahashi, and N. Itoh, 'Digital Modulation Method for Recordable Digital Video Disc', *IEEE Trans. Consumer Electronics.*, vol. 42, pp. 820-825, Aug. 1996.
- [215] A. Kunisa and N. Itoh, 'Digital Modulation and Demodulation', US Patent 6,141,787, Oct. 2000.
- [216] E. Labin and P.R. Asgrain, 'Electric Pulse Communication System', UK Patent 713,614, 1951.
- [217] E.A. Lee and D.G. Messerschmidt, *Digital Communications*, Second Edition, Kluwer Academic Publishers, 1994.
- [218] B.G. Lee and S.C. Kim, *Scrambling Techniques for Digital Transmission*, London UK Springer-Verlag 1994.
- [219] J. Lee, 'Run-length Limited (3,11) Code for High Density Optical Storage Systems', *Electronics Letters.*, vol. 36, No. 9, pp. 810-811, April 2000.
- [220] A. Lempel and M. Cohn, 'Look-Ahead Coding for Input-Restricted Channels', *IEEE Trans. Inform. Theory*, vol. IT-28, no. 6, pp. 933-937, Nov. 1982.
- [221] J. Li and J. Moon, 'DC-Free Run-Length-Limited Codes for Magnetic Recording', *IEEE Trans. Magn.*, vol. 33, no. 1, pp. 868-874, Jan. 1997.

- [222] Y. Lin and P.-H. Liu, 'A Construction Technique for Charge Constrained $(1, k \geq 4)$ Codes', *IEEE Trans. Magn.*, vol. MAG-31, no. 6, pp. 3081-3083, Nov. 1995.
- [223] ———, 'Charge-Constrained $(0, G/I; C)$ Sequences', *IEEE Trans. Commun.*, vol. 45, no. 10, pp. 1183-1191, Oct. 1997.
- [224] D. Lind and B. Marcus, *Symbolic Dynamics and Coding*, Cambridge University Press, 1995.
- [225] Y.X. Li, S.Y. Hsu, and T. Tsao, 'A New Design of EFM Constrained Codes', *IEEE Trans. on Magn.*, vol. MAG-31, no. 6, Nov. 1995
- [226] D.A. Lindholm, 'Power Spectra of Channel Codes for Digital Magnetic Recording', *IEEE Trans. Magn.*, vol. MAG-14, no. 5, pp. 321-323, Sept. 1978.
- [227] N.D. Mackintosh, 'The Choice of a Recording Code', *Proceedings Int. Conf. on Video and Data Recording, IERE Conf. Proc. 43*, Southampton, pp. 77-120, July 1979.
- [228] J.C. Mallinson and J.W. Miller, 'Optimal Codes for Digital Magnetic Recording', *Radio and Elec. Eng.*, vol. 47, pp. 172-176, 1977.
- [229] M. Mansuripur, 'Enumerative Modulation Coding with Arbitrary Constraints and Post-Modulation Error Correction Coding for Data Storage Systems', *Optical Data Storage, SPIE*, pp. 76-86, vol. 1499, 1991.
- [230] M.W. Marcellin and H.J. Weber, 'Two-Dimensional Modulation Codes', *IEEE Journal on Selected Areas in Communications*, vol. 10, no. 1, pp. 254-266, Jan. 1992.
- [231] B.H. Marcus, 'Sofic Systems and Encoding Data', *IEEE Trans. Inform. Theory*, vol. IT-31, no. 3, pp. 366-377, May 1985.
- [232] B.H. Marcus, A.M. Patel, and P.H. Siegel, 'Method and Apparatus for Implementing a PRML Code', *US Patent 4,786,890*, Nov. 1988.
- [233] B.H. Marcus and P.H. Siegel, 'On Codes with Spectral Nulls at Rational Sub-multiples of the Symbol Frequency', *IEEE Trans. Inform. Theory*, vol. IT-33, no. 4, pp. 557-568, July 1987.
- [234] 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. IT-37, no. 3, part 2, pp. 742-758, May 1991.

- [235] B.H. Marcus, P.H. Siegel, and J.K. Wolf, 'Finite-state Modulation Codes for Data Storage', *IEEE Journal on Selected Areas in Communications*, vol. 10, no. 1, pp. 5-37, Jan. 1992.
- [236] B.H. Marcus, R.M. Roth, and P.H. Siegel, 'Constrained Systems and Coding for Recording Channels', in *Handbook of Coding Theory*, R. Brualdi, C. Huffman, and V. Pless, Editors, Amsterdam, The Netherlands, Elsevier Press, 1996.
- [237] M.A. McClellan, 'Runlength-limited Code and Method', US Patent 6,285,302, Sept. 2001.
- [238] S.B. McClelland, 'Compatible Digital Magnetic Recording System', US Patent 4,261,019, April 1981.
- [239] P. McEwen, B. Zafar, and K. Fitzpatrick, 'High Rate Runlength Limited Codes for 8-bit ECC Symbols', US Patent 6,201,485, March 2001.
- [240] S.W. McLaughlin, 'Five Runlength-limited Codes for M -ary Recording Channels', *IEEE Trans. on Mag.*, vol. MAG-33, no. 3, pp. 2442-2450, May 1997
- [241] ———, 'The Construction of M -ary $(d, 8)$ Codes that Achieve Capacity and Have the Fewest Number of Encoder States', *IEEE Trans. on Inform. Theory*, vol. IT-43, no. 2, pp. 699-703, March 1997
- [242] S.W. McLaughlin, J. Luo, and Q. Xie, 'On the Capacity of M -ary Runlength-Limited Codes', *IEEE Trans. Inform. Theory*, vol. IT-41, no. 5, pp. 1508-1511, Sept. 1995.
- [243] D.H. McMahon, A.A. Kirby, B.A. Schofield, and K. Springer, 'Data and Forward Error Control Coding Techniques for Digital Signals', US Patent 5,396,239, March 1995.
- [244] C.D. Mee and E.D. Daniel, *Magnetic Recording*, McGraw-Hill Book Company, New York, 1987.
- [245] C. Menyennett and H.C. Ferreira, 'Sequences and Codes with Asymmetrical Runlength Constraints', *IEEE Trans. Commun.*, vol. COM-43, pp. 1862-1865, May 1995.
- [246] O. Milenkovic and B. Vasic, 'Permutation (d, k) codes: Efficient Enumerative Coding and Phrase Length Distribution Shaping', *IEEE Trans. Inform. Theory*, vol. IT-46, no. 7, pp. 2671-2675, Nov. 2000.
- [247] A. Miller, 'Transmission System', US Patent 3,108,261, Oct. 1963.

- [248] J.W. Miller, 'DC-Free Encoding for Data Transmission System', US Patent 4,027,335, May 1977.
- [249] J.W. Miller and P.J. Rudnick, 'Limited Look-Ahead Means', US Patent 4,437,086, March 1984.
- [250] J. Ming, K.A.S. Immink, and B. Farhang-Boroujeny, 'Design Techniques for Weakly Constrained Codes', *Trans. on Commun.*, vol. 51, no. 5, pp. 709-714, May 2003.
- [251] D.S. Modha and B.H. Marcus, 'Art of Constructing Low-complexity Encoders/decoders for Constrained Block Codes', *IEEE Journal on Selected Areas in Communications*, vol. 19, no. 4, pp. 589-601, April 2001.
- [252] C.M. Monti and G.L. Pierobon, 'Codes with a Multiple Spectral Null at Zero Frequency', *IEEE Trans. Inform. Theory*, vol. IT-35, pp. 463-472, no. 2, March 1989.
- [253] J. Moon and B.J. Brickner, 'Maximum Transition Run Codes for Data Storage Systems', *IEEE Trans. Magn.*, vol. 32, no. 5, pt. 1, pp. 3992-3994, Sept. 1996.
- [254] ———, 'Design of a Rate 6/7 Maximum Transition Run Code', *IEEE Trans. Magn.*, vol. 33, pp. 2749-2751, Sept. 1997.
- [255] ———, 'Method and Apparatus for Implementing Maximum Transition Run Codes', US Patent 5,859,601, Jan. 1999
- [256] Y. Moriyama, 'Data Converting Apparatus', US Patent 5,151,699, Sept. 1992.
- [257] ———, 'Method and Apparatus for Reducing DC Component of RLL Codes', US Patent 5,742,243, April 1998.
- [258] Z. Nagy and K. Zeger, 'Capacity Bounds for the Three-Dimensional (0,1) Runlength Limited Channel', *IEEE Trans. Inform. Theory*, vol. IT-46, pp. 1030-1033, May 2000.
- [259] H. Nakajima and K. Odaka, 'A Rotary-head High-density Digital Audio Tape Recorder', *IEEE Trans. Consumer Electr.*, vol. CE-29, pp. 430-437, Aug. 1983.
- [260] T. Narahara, S. Kobayashi, M. Hattori, Y. Shimpuku, G. van den Enden, J.A. Kahlman, M. van Dijk, and R. Woudenberg, 'Optical Disc System for Digital Video Recording', Proceedings Joint Int. Symposium on Optical Memory and Optical Data Storage, Hawaii, pp. 50-52, July 11-15, 1999.

- [261] G.S. Murdock, 'Apparatus and Method for Providing Direct Current Balanced Code', US Patent 6,351,501, Feb. 2002.
- [262] T. Nishiya, K. Tsukano, T. Hirai, S. Mita, and T. Nara, 'Rate 16/17 Maximum Transition Run (3,11) Code on an EEPRL Channel with an Error-Correcting Post-processor', *IEEE Trans. Magn.*, vol. MAG-35, no. 5, pp. 4378-4386, Sept. 1999.
- [263] C. Noda and Y. Ishizawa, 'Data Encoding Method, Apparatus, and Storage Medium', US Patent 6,559,779, May 2003.
- [264] K. Norris and D.S. Bloomberg, 'Channel Capacity of Charge-Constrained Run-Length Limited Codes', *IEEE Trans. Magn.*, vol. MAG-17, no. 6, pp. 3452-3455, Nov. 1981.
- [265] H. Nyquist, 'Certain Topics in Telegraph Transmission Theory', *Trans. AIEE*, vol. 47, pp. 617-644, Feb. 1928.
- [266] K. Odaka, 'Method and Apparatus for Encoding Binary Data', US Patent 4,456,905, June 1984.
- [267] E.K. Orcutt and M.W. Marcellin, 'Enumerable Multi-Track (d, k) Block Codes', *IEEE Trans. Inform. Theory*, vol. IT-39, pp. 1738-1743, Sept. 1993.
- [268] ———, 'Redundant Multi-Track (d, k) Codes', *IEEE Trans. Inform. Theory*, vol. IT-39, pp. 1744-1750, Sept. 1993.
- [269] E. Ordentlich and R.M. Roth, 'Two-Dimensional Weight-Constrained Codes Through Enumeration Bounds', *IEEE Trans. Inform. Theory*, vol. IT-46, pp. 1292-1301, July 2000
- [270] A. Papoulis, *The Fourier Integral and its Applications*, McGraw-Hill Book Company, Inc., New York, 1962.
- [271] ———, *Probability, Random Variables, and Stochastic Processes*, McGraw-Hill Book Company, Inc., New York, 1965.
- [272] A. Patapoutian, J. Stander; P. McEwen, B. Zafer, and J. Fitzpatrick, 'Rate 32/33, $(d = 0, k = 6)$ Runlength Limited Modulation Code Having Optimized Error propagation', US Patent 6,184,806, Feb. 2001.
- [273] A.M. Patel, 'Data Coding with Stable Base Line for Recording and Transmitting Binary Data', US Patent 3,810,111, May 1974.
- [274] ———, 'Zero-modulation Encoding in Magnetic Recording', *IBM J. Res. Develop.*, vol. 19, pp. 366-378, July 1975.

- [275] ———, 'Improved Encoder and Decoder for a Byte-Oriented Rate 8/9, (0,3) Code', *IBM Techn. Discl. Bul.*, vol. 28, pp. 1938, 1975.
- [276] ———, 'Charge-Constrained Byte-Oriented (0,3) code', *IBM Techn. Discl. Bull.*, vol. 19, no. 7, pp. 2715-2717, Dec. 1976.
- [277] L. Patrovics and K.A.S. Immink, 'Encoding of *dclr*-sequences Using One Weight Set', *IEEE Trans. Inform. Theory*, vol. IT-42, no. 5, pp. 1553-1554, Sept. 1996.
- [278] J.B.H. Peek, 'Communications Aspects of the Compact Disc Digital Audio System', *IEEE Commun. Magazine*, vol. 23, pp. 7-15, Feb. 1985.
- [279] M.G. Pelchat and J.M. Geist, 'Surprising Properties of Two-level "Bandwidth Compaction" Codes', *IEEE Trans. Commun.*, vol. COM-23, pp. 878-883, Sept. 1975.
- [280] ———, Correction to 'Surprising Properties of Two-level "Bandwidth Compaction" Codes', *IEEE Trans. Commun.*, vol. COM-24, pp. 479, April 1976.
- [281] K. Petersen, 'Chains, Entropy, Coding', *Ergod. Theory and Dynam. Syst.*, vol. 6, pp. 415-448, 1987.
- [282] W.W. Peterson and E.J. Weldon, *Error-Correcting Codes*, MIT Press, Cambridge, Massachusetts, 1972.
- [283] B.E. Phelps, 'Magnetic Recording Method', US Patent no. 2,774,646, Dec. 1960.
- [284] G.L. Pierobon, 'Codes for Zero Spectral Density at Zero Frequency', *IEEE Trans. Inform. Theory*, vol. IT-30, no. 2, pp. 435-439, March 1984.
- [285] K.C. Pohlman, *The Compact Disc Handbook*, A-R Editions, Madison, 1992.
- [286] G. Polya, 'Picture Writing', *American Mathematical Monthly*, pp. 689, Dec. 1956,
- [287] A.H. Reeves, 'Electric Signaling System', US Patent 2,272,070, Nov. 1939.
- [288] L. Reggiani and G. Tartara, 'On Reverse Concatenation and Soft Decoding Algorithms for PRML Magnetic Recording Channels', *IEEE Journal on Selected Areas in Communications*, vol. 19, no. 4, pp. 612-618, April 2001.

- [289] G. van Rensburg and H.C. Ferreira, 'Four New Runlength Constrained Recording Codes', *Electronics Letters*, vol. 24, pp. 1110-1111, Aug. 1988.
- [290] J. Riordan, *An Introduction to Combinatorial Analysis*, Princeton University Press, 1980.
- [291] R.M. Roth, 'On Runlength-limited Coding with Dc-control', *IEEE Trans. Commun.*, vol. COM-48, pp. 351-358, March 2000.
- [292] R.M. Roth, P.H. Siegel, and A. Vardy, 'Higher-Order Spectral-Null Codes: Constructions and Bounds', *IEEE Trans. Inform. Theory*, vol. IT-40, pp. 1826-1840, Nov. 1994.
- [293] G. Ruckenstein and R.M. Roth, 'Lower Bounds on the Anticipation of Encoders for Input-constrained Channels', *IEEE Trans. Info. Theory*, vol. IT-47, pp. 1796-1812, July 2001.
- [294] N.R. Saxena and J.P. Robinson, 'Accumulator Compression Testing', *IEEE Trans. Computers*, C-35, pp. 317-321, April 1986
- [295] M.P. Schuetzenberger, 'On the Synchronizing Properties of Certain Prefix Codes', *Information and Control*, vol. 7, pp. 23-36, 1964.
- [296] C.E. Shannon, 'A Mathematical Theory of Communication', *Bell Syst. Tech. J.*, vol. 27, pp. 379-423, July 1948.
- [297] P.D. Shaft, 'Bandwidth Compaction Codes for Communications', *IEEE Trans. Commun.*, vol. COM-21, pp. 687-695, June 1973.
- [298] J. Shim and Y. Won, 'Method of Generating Runlength Limited (RLL) Code Having Improved DC-Suppression Capability and Modulation/Demodulation Method of the Generated RLL Code', US Patent 6,268,810, July 2001.
- [299] N. Shirota, 'Method and Apparatus for Reducing DC Components in a Digital Information Signal', US Patent 4,387,364, June 1983.
- [300] P.H. Siegel, 'Recording Codes for Digital Magnetic Storage', *IEEE Trans. Magn.*, vol. MAG-21, no. 5, pp. 1344-1349, Sept. 1985.
- [301] P.H. Siegel and A. Vardy, 'Method and Apparatus for Constructing Asymptotically Optimal Second Order Dc-free Channel Codes', US Patent 5,450,443, Sept. 1995.
- [302] P.H. Siegel and J.K. Wolf, 'Modulation and Coding for Information Storage', *IEEE Commun. Magazine*, vol. 29, no. 12, pp. 68-86, Dec. 1991.

- [303] ———, 'Bit-stuffing Bounds on the Capacity of Two-Dimensional Constrained Arrays', *Proc. 1998 IEEE Int. Symposium Inform. Theory.*, pp. 323, 1998.
- [304] V. Skachek, T. Etzion, and R.M. Roth, 'Efficient Encoding Algorithm for Third-order Spectral-Null Codes', *IEEE Trans. Inform. Theory*, vol. IT-44, pp. 846-851, March 1998.
- [305] E. Soljanin, 'Low disparity Coding Method for Digital Data', US Patent 6,188,337, Feb. 2001.
- [306] E. Soljanin and C.N. Georghiades, 'Coding for Two-Head Recording Systems', *IEEE Trans. Inform. Theory*, vol. IT-41, no. 3, pp. 794-755, May 1995.
- [307] J.L. Sonntag, 'Apparatus and Method for Increasing Density of Runlength Limited Codes without Increasing Error Propagation', US Patent 5,604,497, Feb. 1997.
- [308] S.G. Stan, *The CD-ROM Drive. A Brief System Description*, Kluwer Academic Publishers, Boston/Dordercht/London, 1998.
- [309] R.W. Stevens, 'Data Transmission Code', US Patent 5,025,256, June 1991.
- [310] J.J. Stiffler, *Theory of Synchronous Communications*, Prentice-Hall, Inc., Engelwood Cliffs, 1971.
- [311] R.D. Swanson and J.K. Wolf, 'A New Class of Two-Dimensional RLL Recording Codes', *IEEE Trans. Magn.*, vol.28, pp. 3407-3416, Nov. 1992.
- [312] N.L. Swenson and J.M. Cioffi, 'Sliding-Block Line Codes to Increase Dispersion-Limited Distance of Optical Fiber Channels', *IEEE Journal on Selected Areas in Communications*, vol. 13, no. 3, pp. 485-498, April 1995.
- [313] L.G. Tallini and B. Bose, 'On Efficient High-Order Spectral-Null Codes', *IEEE Trans. Inform. Theory*, vol. IT-45, no. 7, pp. 2594-2601, Nov. 1999.
- [314] L.G. Tallini, R.M. Capocelli, and B. Bose, 'Design of Some New Balanced Null Codes', *IEEE Trans. Inform. Theory*, vol. IT-42, no. 3, pp. 790-802, May 1996.
- [315] R. Talyansky, T. Etzion, and R.M. Roth, 'Efficient Code Construction for Certain Two-Dimensional Constraints', *IEEE Trans. Inform. Theory*, vol. IT-45, no. 2, pp. 794-799, March 1999.

- [316] S. Tanaka, 'Method and Apparatus for Encoding Binary Data', US Patent 4,728,929, March 1988.
- [317] S. Tanaka, T. Shimada, K. Hirayama, and H. Yamada, 'Single Merging Bit Dc-suppressed Run Length Limited Coding', US Patent 5,774,078, June 1998.
- [318] S. Tanaka, T. Shimada, T. Kojima, K. Moriyama, F. Yokogawa, T. Arai, T. Takeuchi, and T. Takeuchi, 'Digital Modulation Apparatus, a Digital Modulation Method, and a Recording Medium Therefore', US Patent 5,917,857, June 1999.
- [319] 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.
- [320] D.T. Tang, 'Run-length-limited Coding for Modified Raised-cosine Equalization Channel', US Patent 3,647,964, March 1972.
- [321] S. Tazaki, F. Takeda, H. Osawa, and Y. Yamada 'Performance Comparison of 8-10 Conversion Codes', *Proc. 5th Int. Conf. on Video and Data Recording*, Southampton, pp. 79-84, 1984.
- [322] T.J. Tjalkens, 'Runlength limited sequences', *IEEE Trans. Inform. Theory*, vol. IT-40, no. 3, pp. 934-940, 1994.
- [323] S.A. Turk and C.P Zook, 'Selectively Removing Sequences from an Enumerative Modulation Code', US Patent 6,271,776, Aug. 2001.
- [324] T. Uehara, H. Minaguchi, and Y. Oba, 'Digital Modulation Method', US Patent 4,988,999, Jan. 1999.
- [325] R.S Varga, *Matrix Iterative Analysis*, Prentice-Hall, Inc., Engelwood Cliffs, 1962.
- [326] A. Vardy, M. Blaum, P.H. Siegel, and G.T. Sincerbox, 'Conservative Arrays: Multi-Dimensional Modulation Codes for Holographic Recording', *IEEE Trans. Inform. Theory*, vol. IT-42, no. 1, pp. 227-230, Jan. 1996.
- [327] B.V. Vasic, 'Capacity of Channels with Redundant Multi-Track (d, k) constraints: The $k < d$ Case', *IEEE Trans. Inform. Theory*, vol. IT-42, no. 5, pp. 1546-1548, Sept. 1996.
- [328] ———, 'Spectral Analysis of Maximum Entropy Multi-Track Modulation Codes', *IEEE Trans. Inform. Theory*, vol. IT-44, no. 4, pp. 1574-1587, July 1998.

- [329] B.V. Vasic, G. Djordjevic, and M. Tasic, 'Loose Composite Constraint Codes and Their Application in DVD', *IEEE Journal on Selected Areas in Communications*, vol. 19, pp. 665-773, April 2001.
- [330] B.V. Vasic, S.W. McLaughlin, O. Milenkovic, 'Shannon Capacity of M -ary Redundant Multi-Track Runlength Limited Codes', *IEEE Trans. Inform. Theory*, vol. IT-44, no. 2, pp. 766-774, March 1998.
- [331] N.N. Vorob'ev, *Fibonacci Numbers*, Gainesville New Classics Library, 1983.
- [332] Y.H.W. Wang, K.A.S. Immink, X.B. Xu, and C.T. Chong, 'Comparison of Two Coding Schemes for Generating DC-free RLL Sequences', Proceedings Joint Int. Symposium on Optical Memory and Optical Data Storage, Hawaii, pp. 288-290, July 1999.
- [333] J. Watkinson, *The Art of Digital Audio*, Focal Press, London, 1988.
- [334] J. Watkinson, *The D-2 Digital Video Recorder*, Focal Press, London, 1990.
- [335] J. Watkinson, *The Art of Digital Video*, Focal Press, London, 1990.
- [336] A.D. Weathers and R.D. Swanson, 'Apparatus Utilizing a Four-state Encoder for Encoding and Decoding a Sliding Block (1,7) Code', US Patent 5,047,767, Sept. 1991.
- [337] A.D. Weathers and J.K. Wolf, 'A New Rate 2/3 Sliding Block Code for the (1,7) Runlength Constraint with the Minimum Number of Encoder States', *IEEE Trans. Inform. Theory*, vol. IT-37, no. 3, pp. 908-913, May 1991.
- [338] J.H. Weber and K.A.S. Abdel-Ghaffar, 'Cascading Runlength-Limited Sequences', *IEEE Trans. Inform. Theory*, vol. IT-39, no. 6, pp. 1976-1984, Nov. 1993.
- [339] J.L. Westby, '16B/20B encoder', US Patent 5,663,724, Sept. 1997.
- [340] S.B. Wicker and V.K. Bhargava, Edrs, *Reed-Solomon Codes and Their Applications*, IEEE Press, 1994.
- [341] A.X. Widmer, 'Partitioned DC Balanced (0,6), 16B/18B Transmission Code with Error Correction', US Patent 6,198,413, March 2001.
- [342] A.X. Widmer and P.A. Franaszek, 'A Dc-balanced, Partitioned-Block, 8b/10b Transmission Code', *IBM J. Res. Develop.*, vol. 27, no. 5, pp. 440-451, Sept. 1983.

- [343] ———, 'Transmission Code for High-Speed Fiber-Optic Data Networks', *Electronics Letters*, vol. 19, pp. 202-203, 1983.
- [344] ———, 'Byte-oriented DC-balanced (0,4) code 8B/10B Partitioned Block Transmission Code', US Patent 4,486,739, Dec 1984.
- [345] A.J. van Wijngaarden and K.A.S. Immink, 'Construction of Constrained Codes Using Sequence Replacement Techniques', Submitted *IEEE Trans. Inform. Theory*, 1997.
- [346] ———, 'On Guided Scrambling with Guaranteed Maximum Run-length Constraints', IEEE Int. Symposium on Information Theory (ISIT), Chicago, June 2004.
- [347] ———, 'Maximum Runlength Limited Codes with Error Control Capabilities', *IEEE Journal on Selected Areas in Communications*, vol. 19, pp. 602-611, April 2001.
- [348] A.J. van Wijngaarden and E. Soljanin, 'A Combinatorial Technique for Constructing High-rate MTR-RLL Codes', *IEEE Journal on Selected Areas in Communications*, vol. 19, no. 4, pp. 582-588, April 2001.
- [349] ———, 'Methods and Apparatus for Implementing Run-length Limited and Maximum Transition Run Codes', US Patent 6,241,778, June 2001.
- [350] R.W. Wood, 'Magnetic Recording Systems', *Proceedings IEEE*, vol. 74, pp. 1557-1569, Nov. 1986.
- [351] R.W. Wood and D.A. Petersen, 'Viterbi Detection of Class IV Partial Response on a Magnetic Recording Channel', *IEEE Trans. Commun.*, vol. COM-34, pp. 454-461, May 1986.
- [352] Y. Xin and I.J. Fair, 'Algorithms to Enumerate Codewords for DC²-constrained Channels', *IEEE Trans. Inform. Theory*, vol. IT-47, no. 7, pp. 3020-3025, Nov. 2001.
- [353] ———, 'A Performance Metric for Codes with a High-order Spectral at Zero Frequency', *IEEE Trans. Inform. Theory*, vol. IT-50, no. 2, pp. 385-394, Feb. 2004.
- [354] T. Yoshida, 'The Rewritable MiniDisc System', *Proceedings IEEE*, vol. 82, no. 10, pp. 1492-1500, Oct. 1994.
- [355] S. Yoshida and S. Yajima, 'On the Relation Between an Encoding Automaton and the Power Spectrum of its Output Sequences', *Trans. IECE Japan*, vol. 59, pp. 1-7, 1976.

- [356] E. Zehavi, 'Coding for Magnetic Recording', Ph.D. Thesis, University of California, San Diego, 1987.
- [357] E. Zehavi and J.K. Wolf, 'On Runlength Codes', *IEEE Trans. Inform. Theory*, vol. IT-34, no. 1, pp. 45-54, Jan. 1988.

Index

- $(O, G/I)$ sequence, 83
- (dk) sequence, 52
- $(dklr)$ sequence, 106, 108
- (z) sequence, 200
- K th order zero-disparity, 253
- dk constraint, 52
- m -sequence, 260
- n -step transition matrix, 16
- z -constrained sequence, 200
- dc²-balanced, 244

- a.c. coupling, 6
- Abdel-Ghaffar, 313
- Abdel-Khaffar, 109, 284
- ACH algorithm, 160, 172, 284
- adjacency matrix, 21, 60, 88, 120
- Adler, 160, 170, 171, 191, 313
- AES, 309
- Aigrain, 138
- Al-Bassam, 237
- almost block-decodable code, 126
- Alon, 237
- alphabet of symbols, 9, 87
- approximate eigenvector, 100, 170
- Ashley, 62, 63, 172, 192, 228, 313
- asymmetrical constraints, 79
- asymptotic information rate, 56
- audio sample, 114
- auto-correlation function, 30, 75, 244, 249
- auto-covariance function, 30
- automatic gain control(AGC), 273
- autonomous machine, 37, 146
- Aziz, 117, 314

- Bahl, 52, 107, 138
- Baldwin, 181, 285, 314
- bandwidth-limited system, 5, 51
- baseline wander, 6
- Beenker, 106, 315
- Bennett, 32
- Bergmann, 313
- Berkoff, 52, 315
- Bernoulli, 57
- Berstel, 73
- Bertram, 315
- Bhargava, 319
- Biglieri, 315
- binary entropy function, 11
- binary to decimal conversion, 137
- bit stuffing, 163, 275
- Blahut, 12, 315
- Blake, 63, 109, 315
- Blaum, 91, 315
- Bliss, 154, 315
- Bliss' scheme, 155
- block code, 39, 95, 137
- Bloomberg, 278
- BluRay Disc, v, 4, 52, 289, 309
- Bose, 237, 248, 336
- Bosik, 32, 316
- bounded-delay encodable code, 170
- Bowers, 196, 257
- Braat, 316
- Braun, 283, 303, 316
- Brayton, 313
- Brickner, 79, 81
- burst error, 2, 156
- burst error correction, 155

- Calderbank, 154, 319
- Calkin, 91

- Campello, 120
- capacity, 4, 10, 56, 201, 204
- Capocelli, 237
- Cariolaro, 29, 39, 40, 316
- Carter, 196, 316
- Cattermole, 32, 33, 36, 196, 252, 305, 317
- CD-I, v
- CD-ROM, v
- CD-V, v
- change-of-state encoder, 52, 122, 198
- channel capacity, 2, 60, 100, 114, 160, 193, 229
- channel code, v, 2
- channel constraint, vi, 4
- characteristic equation, 17, 56, 79, 204, 278, 279
- characterizing function, 38, 181
- Chien, 200, 229, 317
- Cideciyan, 79, 244, 317
- Cioffi, 336
- Code
 - ($O, G/I$) code, 83
 - k -constraint, 115
 - (0, 1) Code, 172
 - (0, 2) Code, 109
 - (0, 3) Code, 114
 - (0, 6) Code, 116
 - (1, 7) Code, 126, 171
 - (1, 7) parity preserve, 4
 - (1, 8) Code, 167
 - (1, 8) block-decodable code, 126
 - (1, 8) parity preserve, 289
 - (2, 7) Code, 166
 - (2, 9) Code, 132
 - 3PM, 53, 127, 191
 - 8b10b, 231
 - Baldwin, 181, 187
 - Bi-phase, 53
 - bi-phase, 47, 49, 227, 279
 - Delay Modulation, 42
 - EFM, 52, 114, 293, 296
 - EFM15, 301
 - EFM16a, 300
 - EFM16b, 300
 - EFMPlus, 52, 80, 286, 296
 - GCR, 109
 - Immink, 183
 - Manchester, 47
 - MFM, 42, 53, 98, 291
 - Miller, 42, 292
 - Miller-Miller, 292
 - Miller-Squared, 292, 295
 - Reed-Solomon, 2, 154, 296
 - Zero-Modulation, 291
- code, 2
- code efficiency, 96
- code rate, 3, 10
- codebook, 97, 252
- codeword assignment, 97, 115
- Coene, 287, 317
- Cohn, 100
- Compact Disc, v, 52, 114, 293, 309
- compatible EFM, 295
- concatenated scheme, 3
- connection matrix, 21, 60, 88, 200
- constrained channel, 9
- constrained code, 2
- constrained system, 18
- Consumer Electronics, 309
- Copeland, 258, 269
- Coppersmith, 160, 287, 313
- coupling components, 6, 195
- Cover, 138, 318
- crosstalk, 6, 31, 90, 232
- cut-off frequency, 207, 243, 282
- cyclo-stationary, 32, 239
- D1 format, 305
- Daniel, 115
- DAT recorder, 4, 6, 232, 309
- data detection, 83
- Data Storage Institute, 309
- data storage system, 3
- Datta, 90, 140, 318
- dc-balanced, 48

- dc-balanced code, 47, 195
- dc-control, 285, 294, 298
- dc-control bit, 285, 286
- dc-free code, 4, 47, 48, 195
- DCC, 4
- DCRLL, 277
- de-scrambler, 260
- decimal representation, 96
- decoder window, 292, 301
- decomposition, 17
- degenerate pattern, 163
- Delay Modulation, 42
- Deng, 259
- Denissen, 259, 318
- density ratio, 58
- detection window, 58, 82, 129
- deterministic constraint, 10
- difference equation, 56, 202
- digital audio recording, v
- digital sum variation (DSV), 200, 281
- digraph, 14
- directed graph, 14
- discrete components, 41
- discrete noiseless channel, 20
- disparity, 196
- distribution vector, 16
- Dolivo, 317
- DSV (digital sum variation), 200, 281
- DVC, 4, 309
- DVD, v, 4, 284, 296, 309

- ECC, 3, 154, 156
- edge, 14
- edge graph, 18, 174
- EFM alternatives, 300
- Eggenberger, 85, 167, 318
- eigenvalue inequality, 100
- eigenvector, 17
- Eleftheriou, 244, 317
- encoder efficiency, 230
- ensemble average, 30
- entropy, 9, 11, 19, 201
- entropy of Markov sources, 18
- entropy of memoryless source, 10
- enumerative encoding, 126, 129, 137
- equilibrium distribution vector, 16
- ergodic chain, 15, 34
- error burst length, 152
- error propagation, 95, 104, 133, 138, 152, 154, 159, 161, 167, 171, 270
- Etzion, 91, 318
- European Patent Office (EPO), 306
- expectation, 30

- Fair, 196, 247, 257, 318
- Fan, 154, 157, 319
- feedback register, 146
- Feller, 73
- Ferreira, 81, 191, 331
- Fibonacci numbers, 54, 124
- fingerprints, 195, 212
- finite-state machine, 36, 60, 61, 200
- Fisher, 119
- Fitinghof, 154, 319
- Fitzpatrick, 85, 119
- floating point arithmetic, 144
- floppy disk, 4
- focal plane, 6
- focus servo, 6
- Forsberg, 63, 148
- Fourier transform, 31, 46, 254
- frame, 71, 169
- frame synchronization, 71, 169, 260
- Franaszek, 99, 103, 159, 169, 231, 251, 319
- Franks, 32
- Fredrickson, 320
- Freiman, 52, 256
- French, 192, 320
- FSSM, 36
- Fuja, 111, 300, 321
- Fukuda, 232
- fundamental theorem, 2
- Funk, 81
- future-dependent coding, 170

- Gabor, 52, 169
 Gallager, 12
 Gallopoulos, 66, 280, 321
 Gantmacher, 321
 Geist, 66
 generalized Fibonacci numbers, 55
 generating function, 66, 107, 134, 147, 254
 Georghiades, 90, 336
 Gibbs' inequality, 12
 Gilbert, 71
 Gorog, 321
 graph, 14
 Greenstein, 219, 321
 Gregory, 305
 Griffiths, 196
 Gu, 111, 300
 Guibas, 73
 Guida, 43
 guided scrambling, 196, 257, 285

 Hadamard matrix, 270
 Hadamard Transform, 258, 259, 269
 hard disk drive, 171
 Hassner, 160, 171, 189, 313
 Hayami, 301, 322
 HDD, 115
 Hecht, 43
 Heegard, 66, 192, 280, 321
 Henry, 198, 237, 322
 Herro, 259, 318
 high-rate code, 119
 higher-order dc-constrained sequence, 253
 higher-order edge graph, 174
 higher-order null, 244
 Hilden, 167, 323
 Hodges, 167
 Hoholdt, 198
 Hollmann, 125, 126, 170, 193, 238, 323
 holographic recording, 91
 Horiguchi, 169

 Howe, 167, 323
 Howell, 64, 68, 166, 324

 IEEE, vi, 309
 IEEE ISIT, vi
 Immink, 106, 125, 233, 293, 305, 324
 information content, 9
 information rate, 3
 information source, 9
 Information Theory, 11, 309
 Ino, 287
 input set, 36
 input-restricted, 9, 61
 Institute for Experimental Mathematics, vi, 309
 inter-track interference, 90
 interleaving, 119
 intersymbol interference, 5, 51
 inverse rank, 142
 irreducible chain, 15
 Isailovic, 325
 Isozaki, 292

 Jacoby, 127, 171, 191, 326
 Janssen A.J.E.M., 316
 Janssen, A.J.E.M., 211
 jitter, 51
 Jorgensen, 326
 Justesen, 208, 211, 225, 228, 319

 Kahlman, 288, 289, 326
 Kamabe, 41, 192, 326
 Kanota, 259, 260
 Karabed, 172, 244, 292, 314
 Kashyap, 64, 327
 Katayama, 80
 Kato, 91
 Kautz, 52, 138, 328
 Kerpez, 163, 207, 280, 328
 Kim, 127, 191
 Kitchens, 287, 313, 318
 Knudson, 85
 Knuth, 198, 236, 237, 257, 328
 Kobayashi, 169, 305, 328

- Kost, 171
Kuki, 117
Kunisa, 259, 275, 329
- label, 18
labelled directed graph, 27
Labin, 138
Lagrange multipliers, 22
land, vi
LaserVision, 309
lattice diagram, 14
Lee, 81, 191
Lempel, 100
lexicographical index, 139
lexicographical order, 138, 145
Li, 301
Lin, 284, 292, 295
Lind, 172
Lindholm, 43
line code, 3
Litsyn, 256
Liu, 284, 295
liu, 292
look-ahead dc-control, 270
look-ahead encoding, 159, 170
look-ahead span, 170
look-up table, 106, 109, 121
low-disparity, 215
- magnetic recording, 1
magnetization, 1, 5, 52
Mallinson, 292, 330
Mansuripur, 154, 319
Marcellin, 90, 333
Marcus, 83, 85, 111, 157, 172, 192, 305, 314, 331
Markov chain, 13, 64
Markov condition, 14
Markov information source, 18, 21, 29
Markov model, 264
Markov process, 217
mass data storage, 4
matched-spectral-null code, 244
maxentropic sequence, 7, 22, 64, 198
maxentropic source, 22, 204, 277
maximum likelihood detection, 83
maximum runlength, 51
maximum transition run, 79
maximum-length sequence, 261
McClellan, 119, 331
McEwen, 119
McLaughlin, 62, 89, 140, 318, 331
McMahon, 154
Mealy machine, 18, 201
measure of information content, 11
Mee, 115
memoryless source, 10
Menyennett, 81
merging bit, 97, 106, 130, 284
merging rule, 97
merging word, 293
Milenkovic, 140
Miller, 332
Ming, 275
MiniDisc, v
minimum runlength, 51
Modha, 317
modulation code, 2
Moivre, 57
Monti, 244, 253
Moon, 79, 81, 332
Moore machine, 18, 37, 176, 201
Morita, 169
Moriyama, 287
Moussouris, 171, 313
MRDS, 264
MSN (matched-spectral-null) code, 244
MTR constraint, 81
multi-level RLL sequences, 87
multi-mode code, 196, 258
multi-track (d, k) -constrained binary codes, 90
multiple spacing, 81
multiple-spaced RLL sequence, 82
Murdock, 219, 333

- Nagai, 259, 260
- nat(ural unit), 11
- National University of Singapore, 309
- next-state function, 36, 181, 298
- Nishiya, 81
- Noda, 295
- node, 14
- noise, 51
- noiseless capacity, 10, 152
- noiseless channel, 10, 61
- Norris, 278
- notch width, 197, 207, 243
- NRZ, 1, 53, 122
- NRZI, 1, 53, 122, 232, 238, 291
- Nyquist, 6, 333

- O'Reilly, 33, 36, 317
- Odaka, 287
- Odlyzko, 73, 313
- offspring state, 173
- Ogawa, 293, 325
- Orcutt, 91, 333
- Ordentlich, 333
- output function, 18, 36, 298
- output set, 37
- overwrite erasure, 232

- packing density, 58
- Papoulis, 29, 333
- parity byte, 154
- parity preserving word assignment, 52, 287
- partial response, 83, 244
- partition, 246, 254
- Pascal's triangle, 139, 214
- Patapoutian, 120, 333
- Patel, 85, 115, 171, 287, 291, 305, 333
- path, 14
- Patrovics, 142, 257
- PCM, v
- pdf, 30
- peak detection, 53
- Pelchat, 66
- period, 32
- Perrin, 73
- Perron-Frobenius theorem, 22
- phase average, 39, 40
- phase-locked loop, 51
- Phelps, 53
- phrase, 25, 59, 65, 79, 93, 140
- Pierobon, 29, 34, 200, 244, 315
- pilot tone, 4, 195
- pit, vi
- pivot bit, 116, 128
- Pohlman, 334
- point-to-point communication link, 3
- polarity bit, 196, 218, 257
- Polya, 134
- polynomial, 163, 201, 246
- post-modulation, 154
- power series, 134
- power spectral density function, 31, 64, 66, 295
- precoder, 52, 122, 234
- prefix code, 164
- prefix condition, 164
- prefix-synchronized format, 71
- principal state, 99, 164, 216
- PRML, 80
- probability density function, 30
- pseudo random sequence, 5, 261
- Pulse Code Modulation, v
- random drawing algorithm, 269
- random-walk, 200
- rate, 3, 10
- rate efficiency, 229
- RDS, 198, 245, 247, 299
- RDS-constrained sequence, 200
- RDSS, 245, 247
- read clock, 51
- recording code, 2
- recursive elimination technique, 99
- redundancy, 2, 3, 10, 197, 213
- Reeves, v
- Reggiani, 334

- regular chain, 15
- Rensburg van, 191
- repetitive-free, 71
- Riordan, 134, 246, 335
- RLL, vi, 4, 51
- RLL sequence with multiple spacings, 82
- RLL/MS constraint, 82
- Robinson, 248
- Roth, 91, 157, 192, 244, 251, 303, 305, 314, 335
- Ruckenstein, 192
- runlength, vi, 10, 51
- runlength constraint, vi, 9, 51, 95, 137
- runlength distribution, 65
- running digital sum, 196, 198, 263, 277, 299

- Saeki, 117
- Saxena, 248
- Schuetzenberger, 73
- Scoopman, 4
- scrambler, 5, 163, 260
- scrambler polynomial, 163, 269
- self clocking, 51
- self synchronizing scrambler, 260
- self-concatenable, 95
- self-punctuating, 164
- self-synchronizing scrambler, 260
- semaphore codes, 73
- sequence replacement technique, 118
- set-concatenation, 111
- Shaft, 52, 335
- Shannon, 2, 9, 11, 19, 61, 272, 335
- Shannon capacity, 4, 10
- shaping function, 34
- shift register, 167, 173
- Shim, 302
- Shirota, 335
- Siegel, 62–64, 66, 85, 91, 172, 228, 244, 251, 292, 305, 313, 321, 335

- Sincerbox, 91, 337
- Skachek, 256, 336
- sliding-block code, 95, 159
- sliding-block compression, 157
- sliding-block decoder, 161, 292
- SMPTE, 309
- Soljanin, 81, 90, 117, 233, 336
- Sonntag, 119, 336
- source alphabet, 18
- source code, 12, 163
- source data, 3
- spectral lines, 41
- spectral notch, 208, 243
- spectral null, 195
- spectral shaping, 4
- spectrum, 29
- splitting rule, 173
- Stan, 336
- state alphabet, 14
- state diagram, 14, 103, 173
- state merging, 172
- state splitting, 155, 172, 176
- state swapping, 299
- state transition matrix, 16
- state-transition diagram, 14, 15, 38, 65
- stationary, 30
- stationary state probability, 16
- statistically independent sequences, 10
- steady-state probability, 16
- Stiffler, 71, 336
- Stirling's approximation, 214
- stochastic matrix, 14
- successor, 19, 176
- sum variance, 197, 203, 209, 264, 266
- surplus edges, 284
- Swanson, 91, 171, 336
- symbolic dynamics, 172
- synchronization, 71, 115
- synchronous variable length code, 164
- systematic part of a codeword, 119

- Tallini, 237, 248, 336
 Talyansky, 91, 336
 Tanaka, 127, 301
 Tang, 52, 107, 138
 Tartara, 334
 Taylor series, 134
 terminal state, 216
 Tezcan, 258, 269
 Tjalkens, 111, 337
 Toeplitz matrix, 200
 Tolhuizen, vii, 259, 318
 tossing of a coin, 12
 track, 1, 90
 track width, 6
 tracking, 195
 transition matrix, 16
 transition probability matrix, 14
 transition table, 38
 trellis diagram, 14, 216
 Tronca, 29, 316
 Tsinghua University, vii
 Turing Machines, 309
 two-dimensional RLL constraints, 90

 uncertainty, 11, 19
 uncoded symbol, 119
 unifilar source, 18, 176, 204
 unique decodability, 104
 US National Television Academy, 309
 US Patent, vi

 Vardy, 91, 244, 251, 337
 variable-length symbol, 24
 variable-length synchronous code, 162
 Vasic, 91, 140, 299, 337
 vertex, 14
 video disc, 309
 video recorder, 5
 Von Mises, 73

 Wang, 290
 Watkinson, 338
 waveform, 2, 9, 29, 43, 51
 weak constraints, 85
 weakly constrained code, 85, 259
 Weathers, 171, 338
 Weber, 90, 109, 284, 313
 weight of a state, 173
 weighting system, 137
 Weldon, 334
 Westby, 232
 Wicker, 338
 Widmer, 231, 233, 338
 Wiener-Kintchine relation, 31
 Wijngaarden, van, 81, 115, 118, 120, 325
 Wilf, 91, 316
 Wilson, 316
 Winchester drive, 98
 window size, 159, 161
 wireless infrared communications, 81
 Wolf, 64, 91, 171, 172, 192, 305, 320
 Won, 302
 Wood, 339
 worst case performance, 275
 WRDS, 264
 Wyner, 52, 320

 Xie, 62
 Xin, 247

 Ytrehus, 323

 Zeger, 91, 332
 Zehavi, 64, 340
 zero-disparity, 196, 247
 Zook, 337