Foundations and Trends® in
sample
Vol. 1, No 1 (2011) 1–217

DOI: xxxxxx

Source Coding: Part I of Fundamentals of Source and Video Coding

Thomas Wiegand[1] and Heiko Schwarz[2]

[1] *Berlin Institute of Technology and Fraunhofer Institute for Telecommunications — Heinrich Hertz Institute, Germany, thomas.wiegand@tu-berlin.de*
[2] *Fraunhofer Institute for Telecommunications — Heinrich Hertz Institute, Germany, heiko.schwarz@hhi.fraunhofer.de*

Abstract

Digital media technologies have become an integral part of the way we create, communicate, and consume information. At the core of these technologies are source coding methods that are described in this text. Based on the fundamentals of information and rate distortion theory, the most relevant techniques used in source coding algorithms are described: entropy coding, quantization as well as predictive and transform coding. The emphasis is put onto algorithms that are also used in video coding, which will be described in the other text of this two-part monograph.

To our families

Contents

1 Introduction **1**

1.1 The Communication Problem 3
1.2 Scope and Overview of the Text 4
1.3 The Source Coding Principle 5

2 Random Processes **7**

2.1 Probability 8
2.2 Random Variables 9
2.2.1 Continuous Random Variables 10
2.2.2 Discrete Random Variables 11
2.2.3 Expectation 13
2.3 Random Processes 14
2.3.1 Markov Processes 16
2.3.2 Gaussian Processes 18
2.3.3 Gauss-Markov Processes 18
2.4 Summary of Random Processes 19

3 Lossless Source Coding **20**

3.1 Classification of Lossless Source Codes 21
3.2 Variable-Length Coding for Scalars 22
3.2.1 Unique Decodability 22
3.2.2 Entropy 27
3.2.3 The Huffman Algorithm 29
3.2.4 Conditional Huffman Codes 31
3.2.5 Adaptive Huffman Codes 33
3.3 Variable-Length Coding for Vectors 33
3.3.1 Huffman Codes for Fixed-Length Vectors 34
3.3.2 Huffman Codes for Variable-Length Vectors 36
3.4 Elias Coding and Arithmetic Coding 40
3.4.1 Elias Coding 41
3.4.2 Arithmetic Coding 47
3.5 Probability Interval Partitioning Entropy Coding 52
3.6 Comparison of Lossless Coding Techniques 61
3.7 Adaptive Coding 62
3.8 Summary of Lossless Source Coding 64

4 Rate Distortion Theory **66**

4.1 The Operational Rate Distortion Function 67
4.1.1 Distortion 68
4.1.2 Rate 70
4.1.3 Operational Rate Distortion Function 70
4.2 The Information Rate Distortion Function 72
4.2.1 Mutual Information 72
4.2.2 Information Rate Distortion Function 76
4.2.3 Properties of the Rate Distortion Function 80
4.3 The Shannon Lower Bound 81
4.3.1 Differential Entropy 81
4.3.2 Shannon Lower Bound 85
4.4 Rate Distortion Function for Gaussian Sources 89
4.4.1 Gaussian IID Sources 90
4.4.2 Gaussian Sources with Memory 91

4.5 Summary of Rate Distortion Theory 98

5 Quantization 100

5.1 Structure and Performance of Quantizers 101
5.2 Scalar Quantization 104
5.2.1 Scalar Quantization with Fixed-Length Codes 106
5.2.2 Scalar Quantization with Variable-Length Codes 111
5.2.3 High-Rate Operational Distortion Rate Functions 119
5.2.4 Approximation for Distortion Rate Functions 125
5.2.5 Performance Comparison for Gaussian Sources 127
5.2.6 Scalar Quantization for Sources with Memory 129
5.3 Vector Quantization 133
5.3.1 Vector Quantization with Fixed-Length Codes 133
5.3.2 Vector Quantization with Variable-Length Codes 137
5.3.3 The Vector Quantization Advantage 138
5.3.4 Performance and Complexity 142
5.4 Summary of Quantization 144

6 Predictive Coding 146

6.1 Prediction 148
6.2 Linear Prediction 152
6.3 Optimal Linear Prediction 154
6.3.1 One-Step Prediction 156
6.3.2 One-Step Prediction for Autoregressive Processes 158
6.3.3 Prediction Gain 160
6.3.4 Asymptotic Prediction Gain 160
6.4 Differential Pulse Code Modulation (DPCM) 163
6.4.1 Linear Prediction for DPCM 165
6.4.2 Adaptive Differential Pulse Code Modulation 172
6.5 Summary of Predictive Coding 174

7 Transform Coding 176

7.1 Structure of Transform Coding Systems 179
7.2 Orthogonal Block Transforms 180

7.3 Bit Allocation for Transform Coefficients 187
7.3.1 Approximation for Gaussian Sources 188
7.3.2 High-Rate Approximation 190
7.4 The Karhunen Loève Transform (KLT) 191
7.4.1 On the Optimality of the KLT 193
7.4.2 Asymptotic Operational Distortion Rate Function 197
7.4.3 Performance for Gauss-Markov Sources 199
7.5 Signal-Independent Unitary Transforms 200
7.5.1 The Walsh-Hadamard Transform (WHT) 201
7.5.2 The Discrete Fourier Transform (DFT) 201
7.5.3 The Discrete Cosine Transform (DCT) 203
7.6 Transform Coding Example 205
7.7 Summary of Transform Coding 207

8 Summary 209

Acknowledgements 212

References 213

1

Introduction

The advances in source coding technology along with the rapid developments and improvements of network infrastructures, storage capacity, and computing power are enabling an increasing number of multimedia applications. In this text, we will describe and analyze fundamental source coding techniques that are found in a variety of multimedia applications, with the emphasis on algorithms that are used in video coding applications. The present first part of the text concentrates on the description of fundamental source coding techniques, while the second part describes their application in modern video coding.

The application areas of digital video today range from multimedia messaging, video telephony, and video conferencing over mobile TV, wireless and wired Internet video streaming, standard- and high-definition TV broadcasting, subscription and pay-per-view services to personal video recorders, digital camcorders, and optical storage media such as the digital versatile disc (DVD) and Blu-Ray disc. Digital video transmission over satellite, cable, and terrestrial channels is typically based on H.222.0/MPEG-2 systems [37], while wired and wireless real-time conversational services often use H.32x [32, 33, 34] or SIP [64], and mobile transmission systems using the Internet and mo-

bile networks are usually based on RTP/IP [68]. In all these application areas, the same basic principles of video compression are employed.

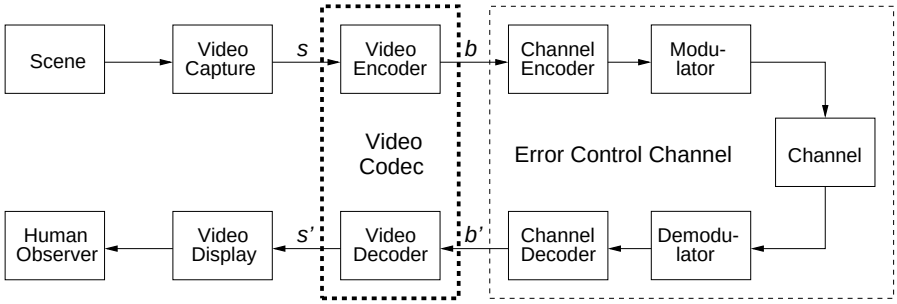


Fig. 1.1 Typical structure of a video transmission system.

The block structure for a typical video transmission scenario is illustrated in Fig. 1.1. The *video capture* generates a video signal s that is discrete in space and time. Usually, the video capture consists of a camera that projects the 3-dimensional *scene* onto an image sensor. Cameras typically generate 25 to 60 frames per second. For the considerations in this text, we assume that the video signal s consists of progressively-scanned pictures. The *video encoder* maps the video signal s into the bitstream b. The bitstream is transmitted over the *error control channel* and the received bitstream b' is processed by the *video decoder* that reconstructs the decoded video signal s' and presents it via the *video display* to the *human observer*. The visual quality of the decoded video signal s' as shown on the video display affects the viewing experience of the human observer. This text focuses on the video encoder and decoder part, which is together called a video codec.

The error characteristic of the digital channel can be controlled by the *channel encoder*, which adds redundancy to the bits at the video encoder output b. The *modulator* maps the channel encoder output to an analog signal, which is suitable for transmission over a physical *channel*. The *demodulator* interprets the received analog signal as a digital signal, which is fed into the *channel decoder*. The channel decoder processes the digital signal and produces the received bitstream b', which may be identical to b even in the presence of channel noise. The sequence of the five components, channel encoder, modula-

tor, channel, demodulator, and channel decoder, are lumped into one box, which is called the error control channel. According to SHANNON's basic work [69, 70] that also laid the ground to the subject of this text, by introducing redundancy at the channel encoder and by introducing delay, the amount of transmission errors can be controlled.

1.1 The Communication Problem

The basic communication problem may be posed as conveying source data with the highest fidelity possible without exceeding an available bit rate, or it may be posed as conveying the source data using the lowest bit rate possible while maintaining a specified reproduction fidelity [69]. In either case, a fundamental trade-off is made between bit rate and signal fidelity. The ability of a source coding system to suitable choose this trade-off is referred to as its *coding efficiency* or *rate distortion performance*. Video codecs are thus primarily characterized in terms of:

- *throughput of the channel*: a characteristic influenced by the transmission channel bit rate and the amount of protocol and error-correction coding overhead incurred by the transmission system;
- *distortion of the decoded video*: primarily induced by the video codec and by channel errors introduced in the path to the video decoder.

However, in practical video transmission systems, the following additional issues must be considered:

- *delay*: a characteristic specifying the start-up latency and end-to-end delay. The delay is influenced by many parameters, including the processing and buffering delay, structural delays of video and channel codecs, and the speed at which data are conveyed through the transmission channel;
- *complexity*: a characteristic specifying the computational complexity, the memory capacity, and memory access requirements. It includes the complexity of the video codec, protocol stacks, and network.

The practical source coding design problem can be stated as follows:

> *Given a maximum allowed delay and a maximum allowed complexity, achieve an optimal trade-off between bit rate and distortion for the range of network environments envisioned in the scope of the applications.*

1.2 Scope and Overview of the Text

This text provides a description of the fundamentals of source and video coding. It is aimed at aiding students and engineers to investigate the subject. When we felt that a result is of fundamental importance to the video codec design problem, we chose to deal with it in greater depth. However, we make no attempt to exhaustive coverage of the subject, since it is too broad and too deep to fit the compact presentation format that is chosen here (and our time limit to write this text). We will also not be able to cover all the possible applications of video coding. Instead our focus is on the source coding fundamentals of video coding. This means that we will leave out a number of areas including implementation aspects of video coding and the whole subject of video transmission and error-robust coding.

The text is divided into two parts. In the first part, the fundamentals of source coding are introduced, while the second part explains their application to modern video coding.

Source Coding Fundamentals. In the present first part, we describe basic source coding techniques that are also found in video codecs. In order to keep the presentation simple, we focus on the description for 1-d discrete-time signals. The extension of source coding techniques to 2-d signals, such as video pictures, will be highlighted in the second part of the text in the context of video coding. Chapter 2 gives a brief overview of the concepts of probability, random variables, and random processes, which build the basis for the descriptions in the following chapters. In Chapter 3, we explain the fundamentals of lossless source coding and present lossless techniques that are found in the video coding area in some detail. The following chapters deal with the topic of lossy compression. Chapter 4 summarizes important

results of rate distortion theory, which builds the mathematical basis for analyzing the performance of lossy coding techniques. Chapter 5 treats the important subject of quantization, which can be considered as the basic tool for choosing a trade-off between transmission bit rate and signal fidelity. Due to its importance in video coding, we will mainly concentrate on the description of scalar quantization. But we also briefly introduce vector quantization in order to show the structural limitations of scalar quantization and motivate the later discussed techniques of predictive coding and transform coding. Chapter 6 covers the subject of prediction and predictive coding. These concepts are found in several components of video codecs. Well-known examples are the motion-compensated prediction using previously coded pictures, the intra prediction using already coded samples inside a picture, and the prediction of motion parameters. In Chapter 7, we explain the technique of transform coding, which is used in most video codecs for efficiently representing prediction error signals.

Application to Video Coding. The second part of the text will describe the application of the fundamental source coding techniques to video coding. We will discuss the basic structure and the basic concepts that are used in video coding and highlight their application in modern video coding standards. Additionally, we will consider advanced encoder optimization techniques that are relevant for achieving a high coding efficiency. The effectiveness of various design aspects will be demonstrated based on experimental results.

1.3 The Source Coding Principle

The present first part of the text describes the fundamental concepts of source coding. We explain various known source coding principles and demonstrate their efficiency based on 1-d model sources. For additional information on information theoretical aspects of source coding the reader is referred to the excellent monographs in [11, 22, 4]. For the overall subject of source coding including algorithmic design questions, we recommend the two fundamental texts by Gersho and Gray [16] and Jayant and Noll [44].

The primary task of a source codec is to represent a signal with the minimum number of (binary) symbols without exceeding an "acceptable level of distortion", which is determined by the application. Two types of source coding techniques are typically named:

- *Lossless coding*: describes coding algorithms that allow the exact reconstruction of the original source data from the compressed data. Lossless coding can provide a reduction in bit rate compared to the original data, when the original signal contains dependencies or statistical properties that can be exploited for data compaction. It is also referred to as *noiseless coding* or *entropy coding*. Lossless coding can only be employed for discrete-amplitude and discrete-time signals. A well-known use for this type of compression for picture and video signals is JPEG-LS [40].
- *Lossy coding*: describes coding algorithms that are characterized by an irreversible loss of information. Only an approximation of the original source data can be reconstructed from the compressed data. Lossy coding is the primary coding type for the compression of speech, audio, picture, and video signals, where an exact reconstruction of the source data is not required. The practically relevant bit rate reduction that can be achieved with lossy source coding techniques is typically more than an order of magnitude larger than that for lossless source coding techniques. Well known examples for the application of lossy coding techniques are JPEG [38] for still picture coding, and H.262/MPEG-2 Video [39] and H.264/AVC [36] for video coding.

Chapter 2 briefly reviews the concepts of probability, random variables, and random processes. Lossless source coding will be described in Chapter 3. The Chapters 5, 6, and 7 give an introduction to the lossy coding techniques that are found in modern video coding applications. In Chapter 4, we provide some important results of rate distortion theory, which will be used for discussing the efficiency of the presented lossy coding techniques.

2

Random Processes

The primary goal of video communication, and signal transmission in general, is the transmission of new information to a receiver. Since the receiver does not know the transmitted signal in advance, the source of information can be modeled as a random process. This permits the description of source coding and communication systems using the mathematical framework of the theory of probability and random processes. If reasonable assumptions are made with respect to the source of information, the performance of source coding algorithms can be characterized based on probabilistic averages. The modeling of information sources as random processes builds the basis for the mathematical theory of source coding and communication.

In this chapter, we give a brief overview of the concepts of probability, random variables, and random processes and introduce models for random processes, which will be used in the following chapters for evaluating the efficiency of the described source coding algorithms. For further information on the theory of probability, random variables, and random processes, the interested reader is referred to [45, 60, 25].

2.1 Probability

Probability theory is a branch of mathematics, which concerns the description and modeling of random events. The basis for modern probability theory is the axiomatic definition of probability that was introduced by KOLMOGOROV in [45] using concepts from set theory.

We consider an experiment with an uncertain outcome, which is called a *random experiment*. The union of all possible *outcomes* ζ of the random experiment is referred to as the *certain event* or *sample space* of the random experiment and is denoted by $\mathcal{O}$. A subset $\mathcal{A}$ of the sample space $\mathcal{O}$ is called an *event*. To each event $\mathcal{A}$ a measure $P(\mathcal{A})$ is assigned, which is referred to as the *probability* of the event $\mathcal{A}$. The measure of probability satisfies the following three axioms:

- Probabilities are non-negative real numbers,
$$P(\mathcal{A}) \geq 0, \qquad \forall \mathcal{A} \subseteq \mathcal{O}. \tag{2.1}$$
- The probability of the certain event $\mathcal{O}$ is equal to 1,
$$P(\mathcal{O}) = 1. \tag{2.2}$$
- The probability of the union of any countable set of pairwise disjoint events is the sum of the probabilities of the individual events; that is, if $\{\mathcal{A}_i : i = 0, 1, \cdots\}$ is a countable set of events such that $\mathcal{A}_i \cap \mathcal{A}_j = \emptyset$ for $i \neq j$, then
$$P\left(\bigcup_i \mathcal{A}_i\right) = \sum_i P(\mathcal{A}_i). \tag{2.3}$$

In addition to the axioms, the notion of the independence of two events and the conditional probability are introduced:

- Two events $\mathcal{A}_i$ and $\mathcal{A}_j$ are *independent* if the probability of their intersection is the product of their probabilities,
$$P(\mathcal{A}_i \cap \mathcal{A}_j) = P(\mathcal{A}_i)\, P(\mathcal{A}_j). \tag{2.4}$$
- The *conditional probability* of an event $\mathcal{A}_i$ given another event $\mathcal{A}_j$, with $P(\mathcal{A}_j) > 0$, is denoted by $P(\mathcal{A}_i|\mathcal{A}_j)$ and is defined as
$$P(\mathcal{A}_i|\mathcal{A}_j) = \frac{P(\mathcal{A}_i \cap \mathcal{A}_j)}{P(\mathcal{A}_j)}. \tag{2.5}$$

The definitions (2.4) and (2.5) imply that, if two events $\mathcal{A}_i$ and $\mathcal{A}_j$ are independent and $P(\mathcal{A}_j) > 0$, the conditional probability of the event $\mathcal{A}_i$ given the event $\mathcal{A}_j$ is equal to the marginal probability of $\mathcal{A}_i$,

$$P(\mathcal{A}_i \,|\, \mathcal{A}_j) = P(\mathcal{A}_i). \tag{2.6}$$

A direct consequence of the definition of conditional probability in (2.5) is Bayes' theorem,

$$P(\mathcal{A}_i|\mathcal{A}_j) = P(\mathcal{A}_j|\mathcal{A}_i)\,\frac{P(\mathcal{A}_i)}{P(\mathcal{A}_j)}, \quad \text{with} \quad P(\mathcal{A}_i),\, P(\mathcal{A}_j) > 0, \tag{2.7}$$

which described the interdependency of the conditional probabilities $P(\mathcal{A}_i|\mathcal{A}_j)$ and $P(\mathcal{A}_j|\mathcal{A}_i)$ for two events $\mathcal{A}_i$ and $\mathcal{A}_j$.

2.2 Random Variables

A concept that we will use throughout this text are *random variables*, which will be denoted with upper-case letters. A random variable S is a function of the sample space $\mathcal{O}$ that assigns a real value $S(\zeta)$ to each outcome $\zeta \in \mathcal{O}$ of a random experiment.

The *cumulative distribution function* (cdf) of a random variable S is denoted by $F_S(s)$ and specifies the probability of the event $\{S \leq s\}$,

$$F_S(s) = P(S \leq s) = P(\,\{\zeta\colon\, S(\zeta) \leq s\}\,). \tag{2.8}$$

The cdf is a non-decreasing function with $F_S(-\infty) = 0$ and $F_S(\infty) = 1$. The concept of defining a cdf can be extended to sets of two or more random variables $\boldsymbol{S} = \{S_0, \cdots, S_{N-1}\}$. The function

$$F_{\boldsymbol{S}}(\boldsymbol{s}) = P(\boldsymbol{S} \leq \boldsymbol{s}) = P(S_0 \leq s_0, \cdots, S_{N-1} \leq s_{N-1}) \tag{2.9}$$

is referred to as *N-dimensional cdf*, *joint cdf*, or *joint distribution*. A set $\boldsymbol{S}$ of random variables is also referred to as a *random vector* and is also denoted using the vector notation $\boldsymbol{S} = (S_0, \cdots, S_{N-1})^T$. For the joint cdf of two random variables X and Y we will use the notation $F_{XY}(x, y) = P(X \leq x, Y \leq y)$. The joint cdf of two random vectors $\boldsymbol{X}$ and $\boldsymbol{Y}$ will be denoted by $F_{\boldsymbol{XY}}(\boldsymbol{x}, \boldsymbol{y}) = P(\boldsymbol{X} \leq \boldsymbol{x}, \boldsymbol{Y} \leq \boldsymbol{y})$.

The *conditional cdf* or *conditional distribution* of a random variable S given an event $\mathcal{B}$, with $P(\mathcal{B}) > 0$, is defined as the conditional

probability of the event $\{S \leq s\}$ given the event $\mathcal{B}$,

$$F_{S|\mathcal{B}}(s \,|\, \mathcal{B}) = P(S \leq s \,|\, \mathcal{B}) = \frac{P(\{S \leq s\} \cap \mathcal{B})}{P(\mathcal{B})}. \tag{2.10}$$

The conditional distribution of a random variable X given another random variable Y is denoted by $F_{X|Y}(x|y)$ and defined as

$$F_{X|Y}(x|y) = \frac{F_{XY}(x,y)}{F_Y(y)} = \frac{P(X \leq x, Y \leq y)}{P(Y \leq y)}. \tag{2.11}$$

Similarly, the conditional cdf of a random vector $\boldsymbol{X}$ given another random vector $\boldsymbol{Y}$ is given by $F_{\boldsymbol{X}|\boldsymbol{Y}}(\boldsymbol{x}|\boldsymbol{y}) = F_{\boldsymbol{XY}}(\boldsymbol{x},\boldsymbol{y})/F_{\boldsymbol{Y}}(\boldsymbol{y})$.

2.2.1 Continuous Random Variables

A random variable S is called a *continuous random variable*, if its cdf $F_S(s)$ is a continuous function. The probability $P(S = s)$ is equal to zero for all values of s. An important function of continuous random variables is the *probability density function* (pdf), which is defined as the derivative of the cdf,

$$f_S(s) = \frac{\mathrm{d}F_S(s)}{\mathrm{d}s} \quad \Leftrightarrow \quad F_S(s) = \int_{-\infty}^{s} f_S(t)\,\mathrm{d}t. \tag{2.12}$$

Since the cdf $F_S(s)$ is a monotonically non-decreasing function, the pdf $f_S(s)$ is greater than or equal to zero for all values of s. Important examples for pdf's, which we will use later in this text, are given below.

Uniform pdf:

$$f_S(s) = 1/A \quad \text{for} \quad -A/2 \leq s \leq A/2, \qquad A > 0 \tag{2.13}$$

Laplacian pdf:

$$f_S(s) = \frac{1}{\sigma_S\sqrt{2}}\, e^{-|s-\mu_S|\sqrt{2}/\sigma_S}, \qquad \sigma_S > 0 \tag{2.14}$$

Gaussian pdf:

$$f_S(s) = \frac{1}{\sigma_S\sqrt{2\pi}}\, e^{-(s-\mu_S)^2/(2\sigma_S^2)}, \qquad \sigma_S > 0 \tag{2.15}$$

The concept of defining a probability density function is also extended to random vectors $\boldsymbol{S} = (S_0, \cdots, S_{N-1})^T$. The multivariate derivative of the joint cdf $F_{\boldsymbol{S}}(\boldsymbol{s})$,

$$f_{\boldsymbol{S}}(\boldsymbol{s}) = \frac{\partial^N F_{\boldsymbol{S}}(\boldsymbol{s})}{\partial s_0 \cdots \partial s_{N-1}}, \tag{2.16}$$

is referred to as the *N-dimensional pdf*, *joint pdf*, or *joint density*. For two random variables X and Y, we will use the notation $f_{XY}(x, y)$ for denoting the joint pdf of X and Y. The joint density of two random vectors $\boldsymbol{X}$ and $\boldsymbol{Y}$ will be denoted by $f_{\boldsymbol{XY}}(\boldsymbol{x}, \boldsymbol{y})$.

The *conditional pdf* or *conditional density* $f_{S|\mathcal{B}}(s|\mathcal{B})$ of a random variable S given an event $\mathcal{B}$, with $P(\mathcal{B}) > 0$, is defined as the derivative of the conditional distribution $F_{S|\mathcal{B}}(s|\mathcal{B})$, $f_{S|\mathcal{B}}(s|\mathcal{B}) = \mathrm{d}F_{S|\mathcal{B}}(s|\mathcal{B})/\mathrm{d}s$. The conditional density of a random variable X given another random variable Y is denoted by $f_{X|Y}(x|y)$ and defined as

$$f_{X|Y}(x|y) = \frac{f_{XY}(x, y)}{f_Y(y)}. \tag{2.17}$$

Similarly, the conditional pdf of a random vector $\boldsymbol{X}$ given another random vector $\boldsymbol{Y}$ is given by $f_{\boldsymbol{X}|\boldsymbol{Y}}(\boldsymbol{x}|\boldsymbol{y}) = f_{\boldsymbol{XY}}(\boldsymbol{x}, \boldsymbol{y})/f_{\boldsymbol{Y}}(\boldsymbol{y})$.

2.2.2 Discrete Random Variables

A random variable S is said to be a *discrete random variable* if its cdf $F_S(s)$ represents a staircase function. A discrete random variable S can only take values of a countable set $\mathcal{A} = \{a_0, a_1, \ldots\}$, which is called the *alphabet* of the random variable. For a discrete random variable S with an alphabet $\mathcal{A}$, the function

$$p_S(a) = P(S = a) = P(\{\zeta\colon S(\zeta) = a\}), \tag{2.18}$$

which gives the probabilities that S is equal to a particular alphabet letter, is referred to as *probability mass function* (pmf). The cdf $F_S(s)$ of a discrete random variable S is given by the sum of the probability masses $p(a)$ with $a \leq s$,

$$F_S(s) = \sum_{a \leq s} p(a). \tag{2.19}$$

With the Dirac delta function δ it is also possible to use a pdf f_S for describing the statistical properties of a discrete random variables S with a pmf $p_S(a)$,

$$f_S(s) = \sum_{a \in \mathcal{A}} \delta(s-a)\, p_S(a). \tag{2.20}$$

Examples for pmf's that will be used in this text are listed below. The pmf's are specified in terms of parameters p and M, where p is a real number in the open interval $(0,1)$ and M is an integer greater than 1. The binary and uniform pmf are specified for discrete random variables with a finite alphabet, while the geometric pmf is specified for random variables with a countably infinite alphabet.

Binary pmf:

$$\mathcal{A} = \{a_0, a_1\}, \qquad p_S(a_0) = p, \qquad p_S(a_1) = 1-p \tag{2.21}$$

Uniform pmf:

$$\mathcal{A} = \{a_0, a_1, \cdots, a_{M-1}\}, \quad p_S(a_i) = 1/M, \qquad \forall\, a_i \in \mathcal{A} \tag{2.22}$$

Geometric pmf:

$$\mathcal{A} = \{a_0, a_1, \cdots\}, \qquad p_S(a_i) = (1-p)\, p^i, \quad \forall\, a_i \in \mathcal{A} \tag{2.23}$$

The pmf for a random vector $\boldsymbol{S} = (S_0, \cdots, S_{N-1})^T$ is defined by

$$p_{\boldsymbol{S}}(\boldsymbol{a}) = P(\boldsymbol{S} = \boldsymbol{a}) = P(S_0 = a_0, \cdots, S_{N-1} = a_{N-1}) \tag{2.24}$$

and is also referred to as *N-dimensional pmf* or *joint pmf*. The joint pmf for two random variables X and Y or two random vectors $\boldsymbol{X}$ and $\boldsymbol{Y}$ will be denoted by $p_{XY}(a_x, a_y)$ or $p_{\boldsymbol{XY}}(\boldsymbol{a_x}, \boldsymbol{a_y})$, respectively.

The *conditional pmf* $p_{S|\mathcal{B}}(a\,|\,\mathcal{B})$ of a random variable S given an event $\mathcal{B}$, with $P(\mathcal{B}) > 0$, specifies the conditional probabilities of the events $\{S = a\}$ given the event B, $p_{S|\mathcal{B}}(a\,|\,\mathcal{B}) = P(S = a\,|\,\mathcal{B})$. The conditional pmf of a random variable X given another random variable Y is denoted by $p_{X|Y}(a_x|a_y)$ and defined as

$$p_{X|Y}(a_x|a_y) = \frac{p_{XY}(a_x, a_y)}{p_Y(a_y)}. \tag{2.25}$$

Similarly, the conditional pmf of a random vector $\boldsymbol{X}$ given another random vector $\boldsymbol{Y}$ is given by $p_{\boldsymbol{X}|\boldsymbol{Y}}(\boldsymbol{a_x}|\boldsymbol{a_y}) = p_{\boldsymbol{XY}}(\boldsymbol{a_x}, \boldsymbol{a_y})/p_{\boldsymbol{Y}}(\boldsymbol{a_y})$.

2.2.3 Expectation

Statistical properties of random variables are often expressed using probabilistic averages, which are referred to as *expectation values* or *expected values*. The expectation value of an arbitrary function $g(S)$ of a continuous random variable S is defined by the integral

$$E\{g(S)\} = \int_{-\infty}^{\infty} g(s)\, f_S(s)\, \mathrm{d}s. \tag{2.26}$$

For discrete random variables S, it is defined as the sum

$$E\{g(S)\} = \sum_{a \in \mathcal{A}} g(a)\, p_S(a). \tag{2.27}$$

Two important expectation values are the *mean* μ_S and the *variance* σ_S^2 of a random variable S, which are given by

$$\mu_S = E\{S\} \qquad \text{and} \qquad \sigma_S^2 = E\big\{(S - \mu_s)^2\big\}. \tag{2.28}$$

For the following discussion of expectation values, we consider continuous random variables. For discrete random variables, the integrals have to be replaced by sums and the pdf's have to be replaced by pmf's.

The expectation value of a function $g(\boldsymbol{S})$ of a set N random variables $\boldsymbol{S} = \{S_0, \cdots, S_{N-1}\}$ is given by

$$E\{g(\boldsymbol{S})\} = \int_{\mathcal{R}^N} g(\boldsymbol{s})\, f_{\boldsymbol{S}}(s)\, \mathrm{d}\boldsymbol{s}. \tag{2.29}$$

The *conditional expectation value* of a function $g(S)$ of a random variable S given an event $\mathcal{B}$, with $P(\mathcal{B}) > 0$, is defined by

$$E\{g(S) \,|\, \mathcal{B}\} = \int_{-\infty}^{\infty} g(s)\, f_{S|\mathcal{B}}(s \,|\, \mathcal{B})\, \mathrm{d}s. \tag{2.30}$$

The conditional expectation value of a function $g(X)$ of random variable X given a particular value y for another random variable Y is specified by

$$E\{g(X) \,|\, y\} = E\{g(X) \,|\, Y = y\} = \int_{-\infty}^{\infty} g(x)\, f_{X|Y}(x, y)\, \mathrm{d}x \tag{2.31}$$

and represents a deterministic function of the value y. If the value y is replaced by the random variable Y, the expression $E\{g(X)|Y\}$ specifies

a new random variable that is a function of the random variable Y. The expectation value $E\{Z\}$ of a random variable $Z = E\{g(X)|Y\}$ can be computed using the iterative expectation rule,

$$\begin{aligned} E\{E\{g(X)|Y\}\} &= \int_{-\infty}^{\infty} \left(\int_{-\infty}^{\infty} g(x)\, f_{X|Y}(x,y)\, \mathrm{d}x \right) f_Y(y)\, \mathrm{d}y \\ &= \int_{-\infty}^{\infty} g(x) \left(\int_{-\infty}^{\infty} f_{X|Y}(x,y)\, f_Y(y)\, \mathrm{d}y \right) \mathrm{d}x \\ &= \int_{-\infty}^{\infty} g(x)\, f_X(x)\, \mathrm{d}x = E\{g(X)\}\,. \end{aligned} \tag{2.32}$$

In analogy to (2.29), the concept of conditional expectation values is also extended to random vectors.

2.3 Random Processes

We now consider a series of random experiments that are performed at time instants t_n, with n being an integer greater than or equal to 0. The outcome of each random experiment at a particular time instant t_n is characterized by a random variable $S_n = S(t_n)$. The series of random variables $\boldsymbol{S} = \{S_n\}$ is called a discrete-time[1] *random process*. The statistical properties of a discrete-time random process $\boldsymbol{S}$ can be characterized by the N-th order joint cdf

$$F_{\boldsymbol{S}_k}(\boldsymbol{s}) = P(\boldsymbol{S}_k^{(N)} \leq \boldsymbol{s}) = P(S_k \leq s_0, \cdots, S_{k+N-1} \leq s_{N-1}). \tag{2.33}$$

Random processes $\boldsymbol{S}$ that represent a series of continuous random variables S_n are called *continuous random processes* and random processes for which the random variables S_n are of discrete type are referred to as *discrete random processes*. For continuous random processes, the statistical properties can also be described by the N-th order joint pdf, which is given by the multivariate derivative

$$f_{\boldsymbol{S}_k}(\boldsymbol{s}) = \frac{\partial^N}{\partial s_0 \, \cdots \, \partial s_{N-1}} F_{\boldsymbol{S}_k}(\boldsymbol{s}). \tag{2.34}$$

[1] Continuous-time random processes are not considered in this text.

For discrete random processes, the N-th order joint cdf $F_{\boldsymbol{S}_k}(\boldsymbol{s})$ can also be specified using the N-th order joint pmf,

$$F_{\boldsymbol{S}_k}(\boldsymbol{s}) = \sum_{\boldsymbol{a} \in \mathcal{A}^N} p_{\boldsymbol{S}_k}(\boldsymbol{a}), \tag{2.35}$$

where $\mathcal{A}^N$ represent the product space of the alphabets $\mathcal{A}_n$ for the random variables S_n with $n = k, \cdots, k+N-1$ and

$$p_{\boldsymbol{S}_k}(\boldsymbol{a}) = P(S_k = a_0, \cdots, S_{k+N-1} = a_{N-1}). \tag{2.36}$$

represents the N-th order joint pmf.

The statistical properties of random processes $\boldsymbol{S} = \{S_n\}$ are often characterized by an N-th order autocovariance matrix $\boldsymbol{C}_N(t_k)$ or an N-th order autocorrelation matrix $\boldsymbol{R}_N(t_k)$. The N-th order autocovariance matrix is defined by

$$\boldsymbol{C}_N(t_k) = E\left\{ \left(\boldsymbol{S}_k^{(N)} - \boldsymbol{\mu}_N(t_k)\right)\left(\boldsymbol{S}_k^{(N)} - \boldsymbol{\mu}_N(t_k)\right)^T \right\}, \tag{2.37}$$

where $\boldsymbol{S}_k^{(N)}$ represents the vector $(S_k, \cdots, S_{k+N-1})^T$ of N successive random variables and $\boldsymbol{\mu}_N(t_k) = E\{\boldsymbol{S}_k^{(N)}\}$ is the N-th order mean. The N-th order autocorrelation matrix is defined by

$$\boldsymbol{R}_N(t_k) = E\left\{ \left(\boldsymbol{S}_k^{(N)}\right)\left(\boldsymbol{S}_k^{(N)}\right)^T \right\}. \tag{2.38}$$

A random process is called *stationary* if its statistical properties are invariant to a shift in time. For stationary random processes, the N-th order joint cdf $F_{\boldsymbol{S}_k}(\boldsymbol{s})$, pdf $f_{\boldsymbol{S}_k}(\boldsymbol{s})$, and pmf $p_{\boldsymbol{S}_k}(\boldsymbol{a})$ are independent of the first time instant t_k and are denoted by $F_{\boldsymbol{S}}(\boldsymbol{s})$, $f_{\boldsymbol{S}}(\boldsymbol{s})$, and $p_{\boldsymbol{S}}(\boldsymbol{a})$, respectively. For the random variables S_n of stationary processes we will often omit the index n and use the notation S.

For stationary random processes, the N-th order mean, the N-th order autocovariance matrix, and the N-th order autocorrelation matrix are independent of the time instant t_k and are denoted by $\boldsymbol{\mu}_N$, $\boldsymbol{C}_N$, and $\boldsymbol{R}_N$, respectively. The N-th order mean $\boldsymbol{\mu}_N$ is a vector with all N elements being equal to the mean μ_S of the random variables S. The N-th order autocovariance matrix $\boldsymbol{C}_N = E\{(\boldsymbol{S}^{(N)} - \boldsymbol{\mu}_N)(\boldsymbol{S}^{(N)} - \boldsymbol{\mu}_N)^T\}$

is a symmetric *Toeplitz matrix*,

$$\boldsymbol{C}_N = \sigma_S^2 \begin{pmatrix} 1 & \rho_1 & \rho_2 & \cdots & \rho_{N-1} \\ \rho_1 & 1 & \rho_1 & \cdots & \rho_{N-2} \\ \rho_2 & \rho_1 & 1 & \cdots & \rho_{N-3} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ \rho_{N-1} & \rho_{N-2} & \rho_{N-3} & \cdots & 1 \end{pmatrix}. \tag{2.39}$$

A Toepliz matrix is a matrix with constant values along all descending diagonals from left to right. For information on the theory and application of Toeplitz matrices the reader is referred to the standard reference [29] and the tutorial [23]. The (k,l)-th element of the autocovariance matrix $\boldsymbol{C}_N$ is given by the autocovariance function $\phi_{k,l} = E\{(S_k - \mu_S)(S_l - \mu_S)\}$. For stationary processes, the autocovariance function depends only on the absolute values $|k-l|$ and can be written as $\phi_{k,l} = \phi_{|k-l|} = \sigma_S^2\, \rho_{|k-l|}$. The N-th order autocorrelation matrix $\boldsymbol{R}_N$ is also is symmetric Toeplitz matrix. The (k,l)-th element of $\boldsymbol{R}_N$ is given by $r_{k,l} = \phi_{k,l} + \mu_S^2$.

A random process $\boldsymbol{S} = \{S_n\}$ for which the random variables S_n are independent is referred to as *memoryless* random process. If a memoryless random process is additionally stationary it is also said to be *independent and identical distributed* (iid), since the random variables S_n are independent and their cdf's $F_{S_n}(s) = P(S_n \leq s)$ do not depend on the time instant t_n. The N-th order cdf $F_{\boldsymbol{S}}(\boldsymbol{s})$, pdf $f_{\boldsymbol{S}}(\boldsymbol{s})$, and pmf $p_{\boldsymbol{S}}(\boldsymbol{a})$ for iid processes, with $\boldsymbol{s} = (s_0, \cdots, s_{N-1})^T$ and $\boldsymbol{a} = (a_0, \cdots, a_{N-1})^T$, are given by the products

$$F_{\boldsymbol{S}}(\boldsymbol{s}) = \prod_{k=0}^{N-1} F_S(s_k), \quad f_{\boldsymbol{S}}(\boldsymbol{s}) = \prod_{k=0}^{N-1} f_S(s_k), \quad p_{\boldsymbol{S}}(\boldsymbol{a}) = \prod_{k=0}^{N-1} p_S(a_k), \tag{2.40}$$

where $F_S(s)$, $f_S(s)$, and $p_S(a)$ are the marginal cdf, pdf, and pmf, respectively, for the random variables S_n.

2.3.1 Markov Processes

A *Markov process* is characterized by the property that future outcomes do not depend on past outcomes, but only on the present outcome,

$$P(S_n \leq s_n \,|\, S_{n-1} = s_{n-1}, \cdots) = P(S_n \leq s_n \,|\, S_{n-1} = s_{n-1}). \tag{2.41}$$

This property can also be expressed in terms of the pdf,

$$f_{S_n}(s_n \mid s_{n-1}, \cdots) = f_{S_n}(s_n \mid s_{n-1}), \tag{2.42}$$

for continuous random processes, or in terms of the pmf,

$$p_{S_n}(a_n \mid a_{n-1}, \cdots) = p_{S_n}(a_n \mid a_{n-1}), \tag{2.43}$$

for discrete random processes,

Given a continuous zero-mean iid process $\boldsymbol{Z} = \{Z_n\}$, a stationary continuous Markov process $\boldsymbol{S} = \{S_n\}$ with mean μ_S can be constructed by the recursive rule

$$S_n = Z_n + \rho\,(S_{n-1} - \mu_S) + \mu_S, \tag{2.44}$$

where ρ, with $|\rho| < 1$, represents the correlation coefficient between successive random variables S_{n-1} and S_n. Since the random variables Z_n are independent, a random variable S_n only depends on the preceding random variable S_{n-1}. The variance σ_S^2 of the stationary Markov process $\boldsymbol{S}$ is given by

$$\sigma_S^2 = E\big\{(S_n - \mu_S)^2\big\} = E\big\{(Z_n - \rho\,(S_{n-1} - \mu_S)\,)^2\big\} = \frac{\sigma_Z^2}{1-\rho^2}, \tag{2.45}$$

where $\sigma_Z^2 = E\big\{Z_n^2\big\}$ denotes the variance of the zero-mean iid process $\boldsymbol{Z}$. The autocovariance function of the process $\boldsymbol{S}$ is given by

$$\phi_{k,l} = \phi_{|k-l|} = E\big\{(S_k - \mu_S)\,(S_l - \mu_S)\big\} = \sigma_S^2\,\rho^{|k-l|}. \tag{2.46}$$

Each element $\phi_{k,l}$ of the N-th order autocorrelation matrix $\boldsymbol{C}_N$ represents a non-negative integer power of the correlation coefficient ρ.

In following chapters, we will often obtain expressions that depend on the determinant $|\boldsymbol{C}_N|$ of the N-th order autocovariance matrix $\boldsymbol{C}_N$. For stationary continuous Markov processes given by (2.44), the determinant $|\boldsymbol{C}_N|$ can be expressed by a simple relationship. Using Laplace's formula, we can expand the determinant of the N-th order autocovariance matrix along the first column,

$$\big|\boldsymbol{C}_N\big| = \sum_{k=0}^{N-1} (-1)^k\,\phi_{k,0}\,\big|\boldsymbol{C}_N^{(k,0)}\big| = \sum_{k=0}^{N-1} (-1)^k\,\sigma_S^2\,\rho^k\,\big|\boldsymbol{C}_N^{(k,0)}\big|, \tag{2.47}$$

where $\boldsymbol{C}_N^{(k,l)}$ represents the matrix that is obtained by removing the k-th row and l-th column from $\boldsymbol{C}_N$. The first row of each matrix $\boldsymbol{C}_N^{(k,l)}$, with $k > 1$, is equal to the second row of the same matrix multiplied by the correlation coefficient ρ. Hence, the first two rows of these matrices are linearly dependent and the determinants $|\boldsymbol{C}_N^{(k,l)}|$, with $k > 1$, are equal to 0. Thus, we obtain

$$\left|\boldsymbol{C}_N\right| = \sigma_S^2 \left|\boldsymbol{C}_N^{(0,0)}\right| - \sigma_S^2\, \rho \left|\boldsymbol{C}_N^{(1,0)}\right|. \tag{2.48}$$

The matrix $\boldsymbol{C}_N^{(0,0)}$ represents the autocovariance matrix $\boldsymbol{C}_{N-1}$ of the order $(N-1)$. The matrix $\boldsymbol{C}_N^{(1,0)}$ is equal to $\boldsymbol{C}_{N-1}$ except that the first row is multiplied by the correlation coefficient ρ. Hence, the determinant $|\boldsymbol{C}_N^{(1,0)}|$ is equal to $\rho\,|\boldsymbol{C}_{N-1}|$, which yields the recursive rule

$$\left|\boldsymbol{C}_N\right| = \sigma_S^2\,(1-\rho^2)\left|\boldsymbol{C}_{N-1}\right|. \tag{2.49}$$

By using the expression $|\boldsymbol{C}_1| = \sigma_S^2$ for the determinant of the 1-st order autocovariance matrix, we obtain the relationship

$$\left|\boldsymbol{C}_N\right| = \sigma_S^{2N}\,(1-\rho^2)^{N-1}. \tag{2.50}$$

2.3.2 Gaussian Processes

A continuous random process $\boldsymbol{S} = \{S_n\}$ is said to be a *Gaussian process* if all finite collections of random variables S_n represent Gaussian random vectors. The N-th order pdf of a stationary Gaussian process $\boldsymbol{S}$ with mean μ_S and variance σ_S^2 is given by

$$f_{\boldsymbol{S}}(\boldsymbol{s}) = \frac{1}{(2\pi)^{N/2}\,|\boldsymbol{C}_N|^{1/2}}\; e^{-\frac{1}{2}(\boldsymbol{s}-\boldsymbol{\mu}_N)^T \boldsymbol{C}_N^{-1}(\boldsymbol{s}-\boldsymbol{\mu}_N)}, \tag{2.51}$$

where $\boldsymbol{s}$ is a vector of N consecutive samples, $\boldsymbol{\mu}_N$ is the N-th order mean (a vector with all N elements being equal to the mean μ_S), and $\boldsymbol{C}_N$ is an N-th order nonsingular autocovariance matrix given by (2.39).

2.3.3 Gauss-Markov Processes

A continuous random process is called a *Gauss-Markov process* if it satisfies the requirements for both Gaussian processes and Markov processes. The statistical properties of a stationary Gauss-Markov are

completely specified by its mean μ_S, its variance σ_S^2, and its correlation coefficient ρ. The stationary continuous process in (2.44) is a stationary Gauss-Markov process if the random variables Z_n of the zero-mean iid process $\boldsymbol{Z}$ have a Gaussian pdf $f_Z(s)$.

The N-th order pdf of a stationary Gauss-Markov process $\boldsymbol{S}$ with the mean μ_S, the variance σ_S^2, and the correlation coefficient ρ is given by (2.51), where the elements $\phi_{k,l}$ of the N-th order autocovariance matrix $\boldsymbol{C}_N$ depend on the variance σ_S^2 and the correlation coefficient ρ and are given by (2.46). The determinant $|\boldsymbol{C}_N|$ of the N-th order autocovariance matrix of a stationary Gauss-Markov process can be written according to (2.50).

2.4 Summary of Random Processes

In this chapter, we gave a brief review of the concepts of random variables and random processes. A random variable is a function of the sample space of a random experiment. It assigns a real value to each possible outcome of the random experiment. The statistical properties of random variables can be characterized by cumulative distribution functions (cdf's), probability density functions (pdf's), probability mass functions (pmf's), or expectation values.

Finite collections of random variables are called random vectors. A countably infinite sequence of random variables is referred to as (discrete-time) random process. Random processes for which the statistical properties are invariant to a shift in time are called stationary processes. If the random variables of a process are independent, the process is said to be memoryless. Random processes that are stationary and memoryless are also referred to as independent and identically distributed (iid) processes. Important models for random processes, which will also be used in this text, are Markov processes, Gaussian processes, and Gauss-Markov processes.

Beside reviewing the basic concepts of random variables and random processes, we also introduced the notations that will be used throughout the text. For simplifying formulas in the following chapters, we will often omit the subscripts that characterize the random variable(s) or random vector(s) in the notations of cdf's, pdf's, and pmf's.

3

Lossless Source Coding

Lossless source coding describes a reversible mapping of sequences of discrete source symbols into sequences of codewords. In contrast to lossy coding techniques, the original sequence of source symbols can be exactly reconstructed from the sequence of codewords. Lossless coding is also referred to as *noiseless coding* or *entropy coding*. If the original signal contains statistical properties or dependencies that can be exploited for data compression, lossless coding techniques can provide a reduction in transmission rate. Basically all source codecs, and in particular all video codecs, include a lossless coding part by which the coding symbols are efficiently represented inside a bitstream.

In this chapter, we give an introduction to lossless source coding. We analyze the requirements for unique decodability, introduce a fundamental bound for the minimum average codeword length per source symbol that can be achieved with lossless coding techniques, and discuss various lossless source codes with respect to their efficiency, applicability, and complexity. For further information on lossless coding techniques, the reader is referred to the overview of lossless compression techniques in [67].

3.1 Classification of Lossless Source Codes

In this text, we restrict our considerations to the practically important case of binary codewords. A codeword is a sequence of binary symbols (bits) of the alphabet $\mathcal{B}=\{0,1\}$. Let $\boldsymbol{S}=\{S_n\}$ be a stochastic process that generates sequences of discrete source symbols. The source symbols s_n are realizations of the random variables S_n. By the process of lossless coding, a message $\boldsymbol{s}^{(L)}=\{s_0,\cdots,s_{L-1}\}$ consisting of L source symbols is converted into a sequence $\boldsymbol{b}^{(K)}=\{b_0,\cdots,b_{K-1}\}$ of K bits.

In practical coding algorithms, a message $\boldsymbol{s}^{(L)}$ is often split into blocks $\boldsymbol{s}^{(N)}=\{s_n,\cdots,s_{n+N-1}\}$ of N symbols, with $1\leq N\leq L$, and a codeword $\boldsymbol{b}^{(\ell)}(\boldsymbol{s}^{(N)})=\{b_0,\cdots,b_{\ell-1}\}$ of ℓ bits is assigned to each of these blocks $\boldsymbol{s}^{(N)}$. The length ℓ of a codeword $\boldsymbol{b}^{\ell}(\boldsymbol{s}^{(N)})$ can depend on the symbol block $\boldsymbol{s}^{(N)}$. The codeword sequence $\boldsymbol{b}^{(K)}$ that represents the message $\boldsymbol{s}^{(L)}$ is obtained by concatenating the codewords $\boldsymbol{b}^{\ell}(\boldsymbol{s}^{(N)})$ for the symbol blocks $\boldsymbol{s}^{(N)}$. A *lossless source code* can be described by the encoder mapping

$$\boldsymbol{b}^{(\ell)}=\gamma\big(\,\boldsymbol{s}^{(N)}\,\big), \tag{3.1}$$

which specifies a mapping from the set of finite length symbol blocks to the set of finite length binary codewords. The decoder mapping

$$\boldsymbol{s}^{(N)}=\gamma^{-1}\big(\,\boldsymbol{b}^{(\ell)}\,\big)=\gamma^{-1}\big(\,\gamma\big(\,\boldsymbol{s}^{(N)}\,\big)\,\big) \tag{3.2}$$

is the inverse of the encoder mapping γ.

Depending on whether the number N of symbols in the blocks $\boldsymbol{s}^{(N)}$ and the number ℓ of bits for the associated codewords are fixed or variable, the following categories can be distinguished:

(1) *Fixed-to-fixed mapping*: A fixed number of symbols is mapped to fixed length codewords. The assignment of a fixed number ℓ of bits to a fixed number N of symbols yields a codeword length of ℓ/N bit per symbol. We will consider this type of lossless source codes as a special case of the next type.

(2) *Fixed-to-variable mapping*: A fixed number of symbols is mapped to variable length codewords. A well-known method for designing fixed-to-variable mappings is the Huffman algorithm for scalars and vectors, which we will describe in sec. 3.2 and sec. 3.3, respectively.

(3) *Variable-to-fixed mapping*: A variable number of symbols is mapped to fixed length codewords. An example for this type of lossless source codes are Tunstall codes [73, 66]. We will not further describe variable-to-fixed mappings in this text, because of its limited use in video coding.

(4) *Variable-to-variable mapping*: A variable number of symbols is mapped to variable length codewords. A typical example for this type of lossless source codes are arithmetic codes, which we will describe in sec. 3.4. As a less-complex alternative to arithmetic coding, we will also present the probability interval projection entropy code in sec. 3.5.

3.2 Variable-Length Coding for Scalars

In this section, we consider lossless source codes that assign a separate codeword to each symbol s_n of a message $\boldsymbol{s}^{(L)}$. It is supposed that the symbols of the message $\boldsymbol{s}^{(L)}$ are generated by a stationary discrete random process $\boldsymbol{S} = \{S_n\}$. The random variables $S_n = S$ are characterized by a finite[1] symbol alphabet $\mathcal{A} = \{a_0, \cdots, a_{M-1}\}$ and a marginal pmf $p(a) = P(S = a)$. The lossless source code associates each letter a_i of the alphabet $\mathcal{A}$ with a binary codeword $\boldsymbol{b}_i = \{b_0^i, \cdots, b_{\ell(a_i)-1}^i\}$ of a length $\ell(a_i) \geq 1$. The goal of the lossless code design is to minimize the average codeword length

$$\bar{\ell} = E\{\ell(S)\} = \sum_{i=0}^{M-1} p(a_i)\, \ell(a_i), \tag{3.3}$$

while ensuring that each message $\boldsymbol{s}^{(L)}$ is *uniquely decodable* given their coded representation $\boldsymbol{b}^{(K)}$.

3.2.1 Unique Decodability

A code is said to be *uniquely decodable* if and only if each valid coded representation $\boldsymbol{b}^{(K)}$ of a finite number K of bits can be produced by only one possible sequence of source symbols $\boldsymbol{s}^{(L)}$.

[1] The fundamental concepts and results shown in this section are also valid for countably infinite symbol alphabets ($M \to \infty$).

A necessary condition for unique decodability is that each letter a_i of the symbol alphabet $\mathcal{A}$ is associated with a different codeword. Codes with this property are called *non-singular codes* and ensure that a single source symbol is unambiguously represented. But if messages with more than one symbol are transmitted, non-singularity is not sufficient to guarantee unique decodability, as will be illustrated in the following.

a_i	$p(a_i)$	code A	code B	code C	code D	code E
a_0	0.5	0	0	0	00	0
a_1	0.25	10	01	01	01	10
a_2	0.125	11	010	011	10	110
a_3	0.125	11	011	111	110	111
$\bar{\ell}$		1.5	1.75	1.75	2.125	1.75

Table 3.1 Example codes for a source with a four letter alphabet and a given marginal pmf.

Table 3.1 shows five example codes for a source with a four letter alphabet and a given marginal pmf. Code A has the smallest average codeword length, but since the symbols a_2 and a_3 cannot be distinguished[2]. Code A is a singular code and is not uniquely decodable. Although code B is a non-singular code, it is not uniquely decodable either, since the concatenation of the letters a_1 and a_0 produces the same bit sequence as the letter a_2. The remaining three codes are uniquely decodable, but differ in other properties. While code D has an average codeword length of 2.125 bit per symbol, the codes C and E have an average codeword length of only 1.75 bit per symbol, which is, as we will show later, the minimum achievable average codeword length for the given source. Beside being uniquely decodable, the codes D and E are also instantaneously decodable, i.e., each alphabet letter can be decoded right after the bits of its codeword are received. The code C does not have this property. If a decoder for the code C receives a bit equal to 0, it has to wait for the next bit equal to 0 before a symbol can be decoded. Theoretically, the decoder might need to wait until the end of the message. The value of the next symbol depends on how many bits equal to 1 are received between the zero bits.

[2] This may be a desirable feature in lossy source coding systems as it helps to reduce the transmission rate, but in this section, we concentrate on lossless source coding. Note that the notation γ is only used for unique and invertible mappings throughout this text.

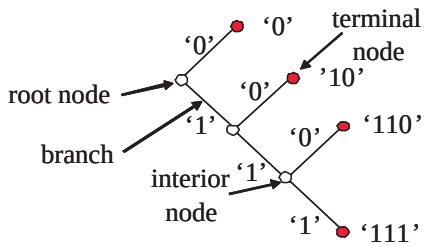


Fig. 3.1 Example for a binary code tree. The represented code is code E of Table 3.1.

Binary Code Trees. Binary codes can be represented using *binary trees* as illustrated in Fig. 3.1. A binary tree is a data structure that consists of *nodes*, with each node having zero, one, or two *descendant nodes*. A node and its descendants nodes are connected by *branches*. A binary tree starts with a *root node*, which is the only node that is not a descendant of any other node. Nodes that are not the root node but have descendants are referred to as *interior nodes*, whereas nodes that do not have descendants are called *terminal nodes* or *leaf nodes*.

In a *binary code tree*, all branches are labeled with '0' or '1'. If two branches depart from the same node, they have different labels. Each node of the tree represents a codeword, which is given by the concatenation of the branch labels from the root node to the considered node. A code for a given alphabet $\mathcal{A}$ can be constructed by associating all terminal nodes and zero or more interior nodes of a binary code tree with one or more alphabet letters. If each alphabet letter is associated with a distinct node, the resulting code is non-singular. In the example of Fig. 3.1, the nodes that represent alphabet letters are filled.

Prefix Codes. A code is said to be a *prefix code* if no codeword for an alphabet letter represents the codeword or a prefix of the codeword for any other alphabet letter. If a prefix code is represented by a binary code tree, this implies that each alphabet letter is assigned to a distinct terminal node, but not to any interior node. It is obvious that every prefix code is uniquely decodable. Furthermore, we will prove later that for every uniquely decodable code there exists a prefix code with exactly the same codeword lengths. Examples for prefix codes are the codes D and E in Table 3.1.

Based on the binary code tree representation the parsing rule for prefix codes can be specified as follows:

(1) Set the current node n_i equal to the root node.
(2) Read the next bit b from the bitstream.
(3) Follow the branch labeled with the value of b from the current node n_i to the descendant node n_j.
(4) If n_j is a terminal node, return the associated alphabet letter and proceed with step 1. Otherwise, set the current node n_i equal to n_j and repeat the previous two steps.

The parsing rule reveals that prefix codes are not only uniquely decodable, but also *instantaneously decodable*. As soon as all bits of a codeword are received, the transmitted symbol is immediately known. Due to this property, it is also possible to switch between different independently designed prefix codes inside a bitstream (i.e., because symbols with different alphabets are interleaved according to a given bitstream syntax) without impacting the unique decodability.

Kraft Inequality. A necessary condition for uniquely decodable codes is given by the Kraft inequality,

$$\sum_{i=0}^{M-1} 2^{-\ell(a_i)} \leq 1. \tag{3.4}$$

For proving this inequality, we consider the term

$$\left(\sum_{i=0}^{M-1} 2^{-\ell(a_i)}\right)^L = \sum_{i_0=0}^{M-1}\sum_{i_1=0}^{M-1}\cdots\sum_{i_{L-1}=0}^{M-1} 2^{-\left(\ell(a_{i_0})+\ell(a_{i_1})+\cdots+\ell(a_{i_{L-1}})\right)}. \tag{3.5}$$

The term $\ell_L = \ell(a_{i_0}) + \ell(a_{i_1}) + \cdots + \ell(a_{i_{L-1}})$ represents the combined codeword length for coding L symbols. Let $A(\ell_L)$ denote the number of distinct symbol sequences that produce a bit sequence with the same length ℓ_L. $A(\ell_L)$ is equal to the number of terms $2^{-\ell_L}$ that are contained in the sum of the right side of (3.5). For a uniquely decodable code, $A(\ell_L)$ must be less than or equal to 2^{ℓ_L}, since there are only 2^{ℓ_L} distinct bit sequences of length ℓ_L. If the maximum length of a codeword is $\ell_{\max}$,

the combined codeword length ℓ_L lies inside the interval $[L, L \cdot \ell_{\max}]$. Hence, a uniquely decodable code must fulfill the inequality

$$\left(\sum_{i=0}^{M-1} 2^{-\ell(a_i)}\right)^L = \sum_{\ell_L=L}^{L\cdot\ell_{\max}} A(\ell_L)\, 2^{-\ell_L} \leq \sum_{\ell_L=L}^{L\cdot\ell_{\max}} 2^{\ell_L}\, 2^{-\ell_L} = L\,(\ell_{\max} - 1) + 1. \tag{3.6}$$

The left side of this inequality grows exponentially with L, while the right side grows only linearly with L. If the Kraft inequality (3.4) is not fulfilled, we can always find a value of L for which the condition (3.6) is violated. And since the constraint (3.6) must be obeyed for all values of $L \geq 1$, this proves that the Kraft inequality specifies a necessary condition for uniquely decodable codes.

The Kraft inequality does not only provide a necessary condition for uniquely decodable codes, it is also always possible to construct a uniquely decodable code for any given set of codeword lengths $\{\ell_0, \ell_1, \cdots, \ell_{M-1}\}$ that satisfies the Kraft inequality. We prove this statement for prefix codes, which represent a subset of uniquely decodable codes. Without loss of generality, we assume that the given codeword lengths are ordered as $\ell_0 \leq \ell_1 \leq \cdots \leq \ell_{M-1}$. Starting with an infinite binary code tree, we chose an arbitrary node of depth ℓ_0 (i.e., a node that represents a codeword of length ℓ_0) for the first codeword and prune the code tree at this node. For the next codeword length ℓ_1, one of the remaining nodes with depth ℓ_1 is selected. A continuation of this procedure yields a prefix code for the given set of codeword lengths, unless we cannot select a node for a codeword length ℓ_i because all nodes of depth ℓ_i have already been removed in previous steps. It should be noted that the selection of a codeword of length ℓ_k removes $2^{\ell_i - \ell_k}$ codewords with a length of $\ell_i \geq \ell_k$. Consequently, for the assignment of a codeword length ℓ_i, the number of available codewords is given by

$$n(\ell_i) = 2^{\ell_i} - \sum_{k=0}^{i-1} 2^{\ell_i - \ell_k} = 2^{\ell_i}\left(1 - \sum_{k=0}^{i-1} 2^{-\ell_k}\right). \tag{3.7}$$

If the Kraft inequality (3.4) is fulfilled, we obtain

$$n(\ell_i) \geq 2^{\ell_i}\left(\sum_{k=0}^{M-1} 2^{-\ell_k} - \sum_{k=0}^{i-1} 2^{-\ell_k}\right) = 1 + \sum_{k=i+1}^{M-1} 2^{-\ell_k} \geq 1. \tag{3.8}$$

Hence, it is always possible to construct a prefix code, and thus a uniquely decodable code, for a given set of codeword lengths that satisfies the Kraft inequality.

The proof shows another important property of prefix codes. Since all uniquely decodable codes fulfill the Kraft inequality and it is always possible to construct a prefix code for any set of codeword lengths that satisfies the Kraft inequality, there do not exist uniquely decodable codes that have a smaller average codeword length than the best prefix code. Due to this property and since prefix codes additionally provide instantaneous decodability and are easy to construct, all variable length codes that are used in practice are prefix codes.

3.2.2 Entropy

Based on the Kraft inequality, we now derive a lower bound for the average codeword length of uniquely decodable codes. The expression (3.3) for the average codeword length $\bar{\ell}$ can be rewritten as

$$\bar{\ell} = \sum_{i=0}^{M-1} p(a_i)\,\ell(a_i) = -\sum_{i=0}^{M-1} p(a_i)\,\log_2\left(\frac{2^{-\ell(a_i)}}{p(a_i)}\right) - \sum_{i=0}^{M-1} p(a_i)\,\log_2 p(a_i). \tag{3.9}$$

With the definition $q(a_i) = 2^{-\ell(a_i)} / \left(\sum_{k=0}^{M-1} 2^{-\ell(a_k)}\right)$, we obtain

$$\bar{\ell} = -\log_2\left(\sum_{i=0}^{M-1} 2^{-\ell(a_i)}\right) - \sum_{i=0}^{M-1} p(a_i)\,\log_2\left(\frac{q(a_i)}{p(a_i)}\right) - \sum_{i=0}^{M-1} p(a_i)\,\log_2 p(a_i). \tag{3.10}$$

Since the Kraft inequality is fulfilled for all uniquely decodable codes, the first term on the right side of (3.10) is greater than or equal to 0. The second term is also greater than or equal to 0 as can be shown using the inequality $\ln x \leq x - 1$ (with equality if and only if $x = 1$),

$$\begin{aligned} -\sum_{i=0}^{M-1} p(a_i)\,\log_2\left(\frac{q(a_i)}{p(a_i)}\right) &\geq \frac{1}{\ln 2}\sum_{i=0}^{M-1} p(a_i)\,\left(1 - \frac{q(a_i)}{p(a_i)}\right) \\ &= \frac{1}{\ln 2}\left(\sum_{i=0}^{M-1} p(a_i) - \sum_{i=0}^{M-1} q(a_i)\right) = 0. \end{aligned} \tag{3.11}$$

The inequality (3.11) is also referred to as divergence inequality for probability mass functions. The average codeword length $\bar{\ell}$ for uniquely decodable codes is bounded by

$$\bar{\ell} \geq H(S) \tag{3.12}$$

with

$$H(S) = E\{-\log_2 p(S)\} = -\sum_{i=0}^{M-1} p(a_i)\, \log_2 p(a_i). \tag{3.13}$$

The lower bound $H(S)$ is called the *entropy* of the random variable S and does only depend on the associated pmf p. Often the entropy of a random variable with a pmf p is also denoted as $H(p)$. The *redundancy* of a code is given by the difference

$$\varrho = \bar{\ell} - H(S) \geq 0. \tag{3.14}$$

The entropy $H(S)$ can also be considered as a measure for the uncertainty[3] that is associated with the random variable S.

The inequality (3.12) is an equality if and only if the first and second term on the right side of (3.10) are equal to 0. This is only the case if the Kraft inequality is fulfilled with equality and $q(a_i) = p(a_i)$, $\forall a_i \in \mathcal{A}$. The resulting conditions $\ell(a_i) = -\log_2 p(a_i)$, $\forall a_i \in \mathcal{A}$, can only hold if all alphabet letters have probabilities that are integer powers of 1/2.

For deriving an upper bound for the minimum average codeword length we choose $\ell(a_i) = \lceil -\log_2 p(a_i) \rceil$, $\forall a_i \in \mathcal{A}$, where $\lceil x \rceil$ represents the smallest integer greater than or equal to x. Since these codeword lengths satisfy the Kraft inequality, as can be shown using $\lceil x \rceil \geq x$,

$$\sum_{i=0}^{M-1} 2^{-\lceil -\log_2 p(a_i) \rceil} \leq \sum_{i=0}^{M-1} 2^{\log_2 p(a_i)} = \sum_{i=0}^{M-1} p(a_i) = 1, \tag{3.15}$$

we can always construct a uniquely decodable code. For the average codeword length of such a code, we obtain, using $\lceil x \rceil < x + 1$,

$$\bar{\ell} = \sum_{i=0}^{M-1} p(a_i)\, \lceil -\log_2 p(a_i) \rceil < \sum_{i=0}^{M-1} p(a_i)\, (1 - \log_2 p(a_i)) = H(S) + 1. \tag{3.16}$$

[3] In SHANNON's original paper [69], the entropy was introduced as an uncertainty measure for random experiments and was derived based on three postulates for such a measure.

The minimum average codeword length $\bar{\ell}_{\min}$ that can be achieved with uniquely decodable codes that assign a separate codeword to each letter of an alphabet always satisfies the inequality

$$H(S) \leq \bar{\ell}_{\min} < H(S) + 1. \tag{3.17}$$

The upper limit is approached for a source with a two-letter alphabet and a pmf $\{p, 1-p\}$ if the letter probability p approaches 0 or 1 [15].

3.2.3 The Huffman Algorithm

For deriving an upper bound for the minimum average codeword length we chose $\ell(a_i) = \lceil -\log_2 p(a_i) \rceil$, $\forall a_i \in \mathcal{A}$. The resulting code has a redundancy $\varrho = \bar{\ell} - H(S_n)$ that is always less than 1 bit per symbol, but it does not necessarily achieve the minimum average codeword length. For developing an optimal uniquely decodable code, i.e., a code that achieves the minimum average codeword length, it is sufficient to consider the class of prefix codes, since for every uniquely decodable code there exists a prefix code with the exactly same codeword length. An optimal prefix code has the following properties:

- For any two symbols $a_i, a_j \in \mathcal{A}$ with $p(a_i) > p(a_j)$, the associated codeword lengths satisfy $\ell(a_i) \leq \ell(a_j)$.
- There are always two codewords that have the maximum codeword length and differ only in the final bit.

These conditions can be proved as follows. If the first condition is not fulfilled, an exchange of the codewords for the symbols a_i and a_j would decrease the average codeword length while preserving the prefix property. And if the second condition is not satisfied, i.e., if for a particular codeword with the maximum codeword length there does not exist a codeword that has the same length and differs only in the final bit, the removal of the last bit of the particular codeword would preserve the prefix property and decrease the average codeword length.

Both conditions for optimal prefix codes are obeyed if two codewords with the maximum length that differ only in the final bit are assigned to the two letters a_i and a_j with the smallest probabilities. In the corresponding binary code tree, a parent node for the two leaf

nodes that represent these two letters is created. The two letters a_i and a_j can then be treated as a new letter with a probability of $p(a_i)+p(a_j)$ and the procedure of creating a parent node for the nodes that represent the two letters with the smallest probabilities can be repeated for the new alphabet. The resulting iterative algorithm was developed and proved to be optimal by Huffman in [30]. Based on the construction of a binary code tree, the Huffman algorithm for a given alphabet $\mathcal{A}$ with a marginal pmf p can be summarized as follows:

(1) Select the two letters a_i and a_j with the smallest probabilities and create a parent node for the nodes that represent these two letters in the binary code tree.
(2) Replace the letters a_i and a_j by a new letter with an associated probability of $p(a_i) + p(a_j)$.
(3) If more than one letter remains, repeat the previous steps.
(4) Convert the binary code tree into a prefix code.

A detailed example for the application of the Huffman algorithm is given in Fig. 3.2. Optimal prefix codes are often generally referred to as *Huffman codes*. It should be noted that there exist multiple optimal prefix codes for a given marginal pmf. A tighter bound than in (3.17) on the redundancy of Huffman codes is provided in [15].

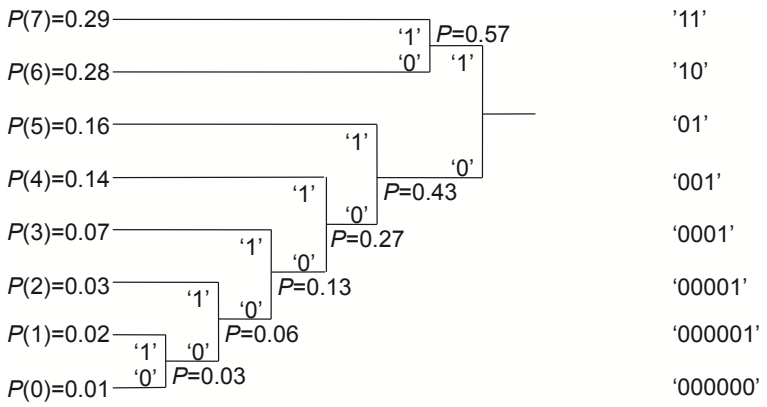


Fig. 3.2 Example for the design of a Huffman code.

3.2.4 Conditional Huffman Codes

Until now, we considered the design of variable length codes for the marginal pmf of stationary random processes. However, for random processes $\{S_n\}$ with memory, it can be beneficial to design variable length codes for conditional pmfs and switch between multiple codeword tables depending on already coded symbols.

a	a_0	a_1	a_2	entropy
$p(a\|a_0)$	0.90	0.05	0.05	$H(S_n\|a_0) = 0.5690$
$p(a\|a_1)$	0.15	0.80	0.05	$H(S_n\|a_1) = 0.8842$
$p(a\|a_2)$	0.25	0.15	0.60	$H(S_n\|a_2) = 1.3527$
$p(a)$	$0.6\overline{4}$	$0.2\overline{4}$	$0.\overline{1}$	$H(S) = 1.2575$

Table 3.2 Conditional pmfs $p(a|a_k)$ and conditional entropies $H(S_n|a_k)$ for an example of a stationary discrete Markov process with a three letter alphabet. The conditional entropy $H(S_n|a_k)$ is the entropy of the conditional pmf $p(a|a_k)$ given the event $\{S_{n-1} = a_k\}$. The resulting marginal pmf $p(a)$ and marginal entropy $H(S)$ are given in the last table row.

As an example, we consider a stationary discrete Markov process with a three symbol alphabet $\mathcal{A} = \{a_0, a_1, a_2\}$. The statistical properties of this process are completely characterized by three conditional pmfs $p(a|a_k) = P(S_n = a \,|\, S_{n-1} = a_k)$ with $k = 0, 1, 2$, which are given in Table 3.2. An optimal prefix code for a given conditional pmf can be designed in exactly the same way as for a marginal pmf. A corresponding Huffman code design for the example Markov source is shown in Table 3.3. For comparison, Table 3.3 lists also a Huffman code for the marginal pmf. The codeword table that is chosen for coding a symbol s_n depends on the value of the preceding symbol s_{n-1}. It is important to note that an independent code design for the conditional pmfs is only possible for instantaneously decodable codes, i.e., for prefix codes.

a_i	Huffman codes for conditional pmfs			Huffman code for marginal pmf
	$S_{n-1} = a_0$	$S_{n-1} = a_2$	$S_{n-1} = a_2$	
a_0	1	00	00	1
a_1	00	1	01	00
a_2	01	01	1	01
$\bar{\ell}$	1.1	1.2	1.4	1.3556

Table 3.3 Huffman codes for the conditional pmfs and the marginal pmf of the Markov process specified in Table 3.2.

The average codeword length $\bar{\ell}_k = \bar{\ell}(S_{n-1}{=}a_k)$ of an optimal prefix code for each of the conditional pmfs is guaranteed to lie in the half-open interval $[H(S_n|a_k), H(S_n|a_k)+1)$, where

$$H(S_n|a_k) = H(S_n|S_{n-1}{=}a_k) = -\sum_{i=0}^{M-1} p(a_i|a_k)\,\log_2 p(a_i|a_k) \tag{3.18}$$

denotes the *conditional entropy* of the random variable S_n given the event $\{S_{n-1} = a_k\}$. The resulting average codeword length $\bar{\ell}$ for the conditional code is

$$\bar{\ell} = \sum_{k=0}^{M-1} p(a_k)\,\bar{\ell}_k. \tag{3.19}$$

The resulting lower bound for the average codeword length $\bar{\ell}$ is referred to as the *conditional entropy* $H(S_n|S_{n-1})$ of the random variable S_n assuming the random variable S_{n-1} and is given by

$$\begin{aligned} H(S_n|S_{n-1}) &= E\{-\log_2 p(S_n|S_{n-1})\} = \sum_{k=0}^{M-1} p(a_k)\,H(S_n|S_{n-1}{=}a_k) \\ &= -\sum_{i=0}^{M-1}\sum_{k=0}^{M-1} p(a_i,a_k)\,\log_2 p(a_i|a_k), \end{aligned} \tag{3.20}$$

where $p(a_i,a_k) = P(S_n{=}a_i, S_{n-1}{=}a_k)$ denotes the joint pmf of the random variables S_n and S_{n-1}. The conditional entropy $H(S_n|S_{n-1})$ specifies a measure for the uncertainty about S_n given the value of S_{n-1}. The minimum average codeword length $\bar{\ell}_{\min}$ that is achievable with the conditional code design is bounded by

$$H(S_n|S_{n-1}) \le \bar{\ell}_{\min} < H(S_n|S_{n-1}) + 1. \tag{3.21}$$

As can be easily shown from the divergence inequality (3.11),

$$\begin{aligned} H(S) - H(S_n|S_{n-1}) &= -\sum_{i=0}^{M-1}\sum_{k=0}^{M-1} p(a_i,a_k)\Big(\log_2 p(a_i) - \log_2 p(a_i|a_k)\Big) \\ &= -\sum_{i=0}^{M-1}\sum_{k=0}^{M-1} p(a_i,a_k)\,\log_2 \frac{p(a_i)\,p(a_k)}{p(a_i,a_k)} \\ &\ge 0, \end{aligned} \tag{3.22}$$

the conditional entropy $H(S_n|S_{n-1})$ is always less than or equal to the marginal entropy $H(S)$. Equality is obtained if $p(a_i, a_k) = p(a_i)p(a_k)$, $\forall a_i, a_k \in \mathcal{A}$, i.e., if the stationary process $\boldsymbol{S}$ is an iid process.

For our example, the average codeword length of the conditional code design is 1.1578 bit per symbol, which is about 14.6% smaller than the average codeword length of the Huffman code for the marginal pmf.

For sources with memory that do not satisfy the Markov property, it can be possible to further decrease the average codeword length if more than one preceding symbol is used in the condition. However, the number of codeword tables increases exponentially with the number of considered symbols. To reduce the number of tables, the number of outcomes for the condition can be partitioned into a small number of events, and for each of these events, a separate code can be designed. As an application example, the CAVLC design in the H.264/AVC video coding standard [36] includes conditional variable length codes.

3.2.5 Adaptive Huffman Codes

In practice, the marginal and conditional pmfs of a source are usually not known and sources are often nonstationary. Conceptually, the pmf(s) can be simultaneously estimated in encoder and decoder and a Huffman code can be redesigned after coding a particular number of symbols. This would, however, tremendously increase the complexity of the coding process. A fast algorithm for adapting Huffman codes was proposed by Gallager [15]. But even this algorithm is considered as too complex for video coding application, so that adaptive Huffman codes are rarely used in this area.

3.3 Variable-Length Coding for Vectors

Although scalar Huffman codes achieve the smallest average codeword length among all uniquely decodable codes that assign a separate codeword to each letter of an alphabet, they can be very inefficient if there are strong dependencies between the random variables of a process. For sources with memory, the average codeword length per symbol can be decreased if multiple symbols are coded jointly. Huffman codes that assign a codeword to a block of two or more successive symbols are

referred to as *block Huffman codes* or *vector Huffman codes* and represent an alternative to conditional Huffman codes[4]. The joint coding of multiple symbols is also advantageous for iid processes for which one of the probabilities masses is close to one.

3.3.1 Huffman Codes for Fixed-Length Vectors

We consider stationary discrete random sources $\boldsymbol{S} = \{S_n\}$ with an M-ary alphabet $\mathcal{A} = \{a_0, \cdots, a_{M-1}\}$. If N symbols are coded jointly, the Huffman code has to be designed for the joint pmf

$$p(a_0, \cdots, a_{N-1}) = P(S_n = a_0, \cdots, S_{n+N-1} = a_{N-1})$$

of a block of N successive symbols. The average codeword length $\bar{\ell}_{\min}$ per symbol for an optimum block Huffman code is bounded by

$$\frac{H(S_n, \cdots, S_{n+N-1})}{N} \leq \bar{\ell}_{\min} < \frac{H(S_n, \cdots, S_{n+N-1})}{N} + \frac{1}{N}, \tag{3.23}$$

where

$$H(S_n, \cdots, S_{n+N-1}) = E\{-\log_2 p(S_n, \cdots, S_{n+N-1})\} \tag{3.24}$$

is referred to as the *block entropy* for a set of N successive random variables $\{S_n, \cdots, S_{n+N-1}\}$. The limit

$$\bar{H}(\boldsymbol{S}) = \lim_{N\to\infty} \frac{H(S_0, \cdots, S_{N-1})}{N} \tag{3.25}$$

is called the *entropy rate* of a source $\boldsymbol{S}$. It can be shown that the limit in (3.25) always exists for stationary sources [14]. The entropy rate $\bar{H}(\boldsymbol{S})$ represents the greatest lower bound for the average codeword length $\bar{\ell}$ per symbol that can be achieved with lossless source coding techniques,

$$\bar{\ell} \geq \bar{H}(\boldsymbol{S}). \tag{3.26}$$

For iid processes, the entropy rate

$$\begin{aligned} \bar{H}(\boldsymbol{S}) &= \lim_{N\to\infty} \frac{E\{-\log_2 p(S_0, S_1, \cdots, S_{N-1})\}}{N} \\ &= \lim_{N\to\infty} \frac{\sum_{n=0}^{N-1} E\{-\log_2 p(S_n)\}}{N} = H(S) \end{aligned} \tag{3.27}$$

[4] The concepts of conditional and block Huffman codes can also be combined by switching codeword tables for a block of symbols depending on the values of already coded symbols.

is equal to the marginal entropy $H(S)$. For stationary Markov processes, the entropy rate

$$\begin{aligned}\bar{H}(\boldsymbol{S}) &= \lim_{N\to\infty} \frac{E\{-\log_2 p(S_0, S_1, \cdots, S_{N-1})\}}{N} \\ &= \lim_{N\to\infty} \frac{E\{-\log_2 p(S_0)\} + \sum_{n=1}^{N-1} E\{-\log_2 p(S_n|S_{n-1})\}}{N} \\ &= H(S_n|S_{n+1}) \qquad (3.28)\end{aligned}$$

is equal to the conditional entropy $H(S_n|S_{n-1})$.

(a)

$a_i a_k$	$p(a_i, a_k)$	codewords
a_0a_0	0.58	1
a_0a_1	$0.03\overline{2}$	00001
a_0a_2	$0.03\overline{2}$	00010
a_1a_0	$0.03\overline{6}$	0010
a_1a_1	$0.19\overline{5}$	01
a_1a_2	$0.01\overline{2}$	000000
a_2a_0	$0.02\overline{7}$	00011
a_2a_1	$0.01\overline{7}$	000001
a_2a_2	$0.0\overline{6}$	0011

(b)

N	$\bar{\ell}$	N_C
1	1.3556	3
2	1.0094	9
3	0.9150	27
4	0.8690	81
5	0.8462	243
6	0.8299	729
7	0.8153	2187
8	0.8027	6561
9	0.7940	19683

Table 3.4 Block Huffman codes for the Markov source specified in Table 3.2: (a) Huffman code for a block of 2 symbols; (b) Average codeword lengths $\bar{\ell}$ and number N_C of codewords depending on the number N of jointly coded symbols.

As an example for the design of block Huffman codes, we consider the discrete Markov process specified in Table 3.2. The entropy rate $\bar{H}(\boldsymbol{S})$ for this source is 0.7331 bit per symbol. Table 3.4(a) shows a Huffman code for the joint coding of 2 symbols. The average codeword length per symbol for this code is 1.0094 bit per symbol, which is smaller than the average codeword length obtained with the Huffman code for the marginal pmf and the conditional Huffman code that we developed in sec. 3.2. As shown in Table 3.4(b), the average codeword length can be further reduced by increasing the number N of jointly coded symbols. If N approaches infinity, the average codeword length per symbol for the block Huffman code approaches the entropy rate. However, the number N_C of codewords that must be stored in an encoder and decoder grows exponentially with the number N of jointly coded symbols. In practice, block Huffman codes are only used for a small number of symbols with small alphabets.

In general, the number of symbols in a message is not a multiple of the block size N. The last block of source symbols may contain less than N symbols, and, in that case, it cannot be represented with the block Huffman code. If the number of symbols in a message is known to the decoder (e.g., because it is determined by a given bitstream syntax), an encoder can send the codeword for any of the letter combinations that contain the last block of source symbols as a prefix. At the decoder side, the additionally decoded symbols are discarded. If the number of symbols that are contained in a message cannot be determined in the decoder, a special symbol for signaling the end of a message can be added to the alphabet.

3.3.2 Huffman Codes for Variable-Length Vectors

An additional degree of freedom for designing Huffman codes, or generally variable-length codes, for symbol vectors is obtained if the restriction that all codewords are assigned to symbol blocks of the same size is removed. Instead, the codewords can be assigned to sequences of a variable number of successive symbols. Such a code is also referred to as V2V code in this text. In order to construct a V2V code, a set of letter sequences with a variable number of letters is selected and a codeword is associated with each of these letter sequences. The set of letter sequences has to be chosen in a way that each message can be represented by a concatenation of the selected letter sequences. An exception is the end of a message, for which the same concepts as for block Huffman codes (see above) can be used.

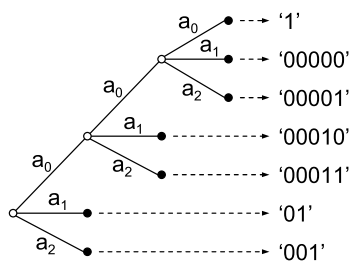


Fig. 3.3 Example for an M-ary tree representing sequences of a variable number of letters, of the alphabet $\mathcal{A} = \{a_0, a_1, a_2\}$, with an associated variable length code.

Similarly as for binary codes, the set of letter sequences can be represented by an M-ary tree as depicted in Fig. 3.3. In contrast to binary code trees, each node has up to M descendants and each branch is labeled with a letter of the M-ary alphabet $\mathcal{A} = \{a_0, a_1, \cdots, a_{M-1}\}$. All branches that depart from a particular node are labeled with different letters. The letter sequence that is represented by a particular node is given by a concatenation of the branch labels from the root node to the particular node. An M-ary tree is said to be a *full tree* if each node is either a leaf node or has exactly M descendants.

We constrain our considerations to full M-ary trees for which all leaf nodes and only the leaf nodes are associated with codewords. This restriction yields a V2V code that fulfills the necessary condition stated above and has additionally the following useful properties:

- *Redundancy-free set of letter sequences*: None of the letter sequences can be removed without violating the constraint that each symbol sequence must be representable using the selected letter sequences.
- *Instantaneously encodable codes*: A codeword can be sent immediately after all symbols of the associated letter sequence have been received.

The first property implies that any message can only be represented by a single sequence of codewords. The only exception is that, if the last symbols of a message do not represent a letter sequence that is associated with a codeword, one of multiple codewords can be selected as discussed above.

Let $N_{\mathcal{L}}$ denote the number of leaf nodes in a full M-ary tree $\mathcal{T}$. Each leaf node $\mathcal{L}_k$ represents a sequence $\boldsymbol{a}_k = \{a_0^k, a_1^k, \cdots, a_{N_k-1}^k\}$ of N_k alphabet letters. The associated probability $p(\mathcal{L}_k)$ for coding a symbol sequence $\{S_n, \cdots, S_{n+N_k-1}\}$ is given by

$$p(\mathcal{L}_k) = p(a_0^k \,|\, \mathcal{B})\; p(a_1^k \,|\, a_0^k,\, \mathcal{B}) \;\cdots\; p(a_{N_k-1}^k \,|\, a_0^k,\, \cdots,\, a_{N_k-2}^k,\, \mathcal{B}), \tag{3.29}$$

where $\mathcal{B}$ represents the event that the preceding symbols $\{S_0, \ldots, S_{n-1}\}$ were coded using a sequence of complete codewords of the V2V tree. The term $p(a_m \,|\, a_0, \cdots, a_{m-1}, \mathcal{B})$ denotes the conditional pmf for a ran-

dom variable S_{n+m} given the random variables S_n to S_{n+m-1} and the event $\mathcal{B}$. For iid sources, the probability $p(\mathcal{L}_k)$ for a leaf node $\mathcal{L}_k$ simplifies to

$$p(\mathcal{L}_k) = p(a_0^k)\, p(a_1^k)\, \cdots\, p(a_{N_k-1}^k). \tag{3.30}$$

For stationary Markov sources, the probabilities $p(\mathcal{L}_k)$ are given by

$$p(\mathcal{L}_k) = p(a_0^k \,|\, \mathcal{B})\, p(a_1^k \,|\, a_0^k)\, \cdots\, p(a_{N_k-1}^k \,|\, a_{N_k-2}^k). \tag{3.31}$$

The conditional pmfs $p(a_m \,|\, a_0, \cdots, a_{m-1}, \mathcal{B})$ are given by the structure of the M-ary tree $\mathcal{T}$ and the conditional pmfs $p(a_m \,|\, a_0, \cdots, a_{m-1})$ for the random variables S_{n+m} assuming the preceding random variables S_n to S_{n+m-1}.

As an example, we show how the pmf $p(a|\mathcal{B}) = P(S_n = a|\mathcal{B})$ that is conditioned on the event $\mathcal{B}$ can be determined for Markov sources. In this case, the probability $p(a_m|\mathcal{B}) = P(S_n = a_m|\mathcal{B})$ that a codeword is assigned to a letter sequence that starts with a particular letter a_m of the alphabet $\mathcal{A} = \{a_0, a_1, \cdots, a_{M-1}\}$ is given by

$$p(a_m|\mathcal{B}) = \sum_{k=0}^{N_{\mathcal{L}}-1} p(a_m|a_{N_k-1}^k)\, p(a_{N_k-1}^k|a_{N_k-2}^k)\, \cdots\, p(a_1^k|a_0^k)\, p(a_0^k|\mathcal{B}). \tag{3.32}$$

These M equations form a homogeneous linear equation system that has one set of non-trivial solutions $p(a|\mathcal{B}) = \kappa \cdot \{x_0, x_1, \cdots, x_{M-1}\}$. The scale factor κ and thus the pmf $p(a|\mathcal{B})$ can be uniquely determined by using the constraint $\sum_{m=0}^{M-1} p(a_m|\mathcal{B}) = 1$.

After the conditional pmfs $p(a_m \,|\, a_0, \cdots, a_{m-1}, \mathcal{B})$ have been determined, the pmf $p(\mathcal{L})$ for the leaf nodes can be calculated. An optimal prefix code for the selected set of letter sequences, which is represented by the leaf nodes of a full M-ary tree $\mathcal{T}$, can be designed using the Huffman algorithm for the pmf $p(\mathcal{L})$. Each leaf node $\mathcal{L}_k$ is associated with a codeword of ℓ_k bits. The average codeword length per symbol $\bar{\ell}$ is given by the ratio of the average codeword length per letter sequence and the average number of letters per letter sequence,

$$\bar{\ell} = \frac{\sum_{k=0}^{N_{\mathcal{L}}-1} p(\mathcal{L}_k)\, \ell_k}{\sum_{k=0}^{N_{\mathcal{L}}-1} p(\mathcal{L}_k)\, N_k}. \tag{3.33}$$

For selecting the set of letter sequences or the full M-ary tree $\mathcal{T}$, we assume that the set of applicable V2V codes for an application is given by parameters such as the maximum number of codewords (number of leaf nodes). Given such a finite set of full M-ary trees, we can select the full M-ary tree $\mathcal{T}$, for which the Huffman code yields the smallest average codeword length per symbol $\bar{\ell}$.

(a)

$\boldsymbol{a}_k$	$p(\mathcal{L}_k)$	codewords
a_0a_0	0.5799	1
a_0a_1	0.0322	00001
a_0a_2	0.0322	00010
a_1a_0	0.0277	00011
$a_1a_1a_0$	0.0222	000001
$a_1a_1a_1$	0.1183	001
$a_1a_1a_2$	0.0074	0000000
a_1a_2	0.0093	0000001
a_2	0.1708	01

(b)

$N_{\mathcal{C}}$	$\bar{\ell}$
5	1.1784
7	1.0551
9	1.0049
11	0.9733
13	0.9412
15	0.9293
17	0.9074
19	0.8980
21	0.8891

Table 3.5 V2V codes for the Markov source specified in Table 3.2: (a) V2V code with $N_{\mathcal{C}} = 9$ codewords; (b) Average codeword lengths $\bar{\ell}$ depending on the number of codewords $N_{\mathcal{C}}$.

As an example for the design of a V2V Huffman code, we again consider the stationary discrete Markov source specified in Table 3.2. Table 3.5(a) shows a V2V code that minimizes the average codeword length per symbol among all V2V codes with up to 9 codewords. The average codeword length is 1.0049 bit per symbol, which is about 0.4% smaller than the average codeword length for the block Huffman code with the same number of codewords. As indicated in Table 3.5(b), when increasing the number of codewords, the average codeword length for V2V codes usually decreases faster as for block Huffman codes. The V2V code with 17 codewords has already an average codeword length that is smaller than that of the block Huffman code with 27 codewords.

An application example of V2V codes is the run-level coding of transform coefficients in MPEG-2 Video [39]. An often used variation of V2V codes is called *run-length coding*. In run-length coding, the number of successive occurrences of a particular alphabet letter, referred to as *run*, is transmitted using a variable-length code. In some applications, only runs for the most probable alphabet letter (including runs equal to 0) are transmitted and are always followed by a codeword for one of

the remaining alphabet letters. In other applications, the codeword for a run is followed by a codeword specifying the alphabet letter, or vice versa. V2V codes are particularly attractive for binary iid sources. As we will show in sec. 3.5, a universal lossless source coding concept can be designed using V2V codes for binary iid sources in connection with the concepts of binarization and probability interval partitioning.

3.4 Elias Coding and Arithmetic Coding

Huffman codes achieve the minimum average codeword length among all uniquely decodable codes that assign a separate codeword to each element of a given set of alphabet letters or letter sequences. However, if the pmf for a symbol alphabet contains a probability mass that is close to one, a Huffman code with an average codeword length close to the entropy rate can only be constructed if a large number of symbols is coded jointly. Such a block Huffman code does however require a huge codeword table and is thus impractical for real applications. Additionally, a Huffman code for fixed- or variable-length vectors is not applicable or at least very inefficient for symbol sequences in which symbols with different alphabets and pmfs are irregularly interleaved, as it is often found in image and video coding applications, where the order of symbols is determined by a sophisticated syntax.

Furthermore, the adaptation of Huffman codes to sources with unknown or varying statistical properties is usually considered as too complex for real-time applications. It is desirable to develop a code construction method that is capable of achieving an average codeword length close to the entropy rate, but also provides a simple mechanism for dealing with nonstationary sources and is characterized by a complexity that increases linearly with the number of coded symbols.

The popular method of *arithmetic coding* provides these properties. The initial idea is attributed to P. Elias (as reported in [1]) and is also referred to as *Elias coding*. The first practical arithmetic coding schemes have been published by Pasco [61] and Rissanen [63]. In the following, we first present the basic concept of Elias coding and continue with highlighting some aspects of practical implementations. For further details, the interested reader is referred to [78], [58] and [65].

3.4.1 Elias Coding

We consider the coding of symbol sequences $\boldsymbol{s} = \{s_0, s_1, \ldots, s_{N-1}\}$ that represent realizations of a sequence of discrete random variables $\boldsymbol{S} = \{S_0, S_1, \ldots, S_{N-1}\}$. The number N of symbols is assumed to be known to both encoder and decoder. Each random variable S_n can be characterized by a distinct M_n-ary alphabet $\mathcal{A}_n$. The statistical properties of the sequence of random variables $\boldsymbol{S}$ are completely described by the joint pmf

$$p(\boldsymbol{s}) = P(\boldsymbol{S}=\boldsymbol{s}) = P(S_0=s_0, S_1=s_1, \cdots, S_{N-1}=s_{N-1}).$$

A symbol sequence $\boldsymbol{s}_a = \{s_0^a, s_1^a, \cdots, s_{N-1}^a\}$ is considered to be less than another symbol sequence $\boldsymbol{s}_b = \{s_0^b, s_1^b, \cdots, s_{N-1}^b\}$ if and only if there exists an integer n, with $0 \leq n \leq N-1$, so that

$$s_k^a = s_k^b \quad \text{for} \quad k = 0, \cdots, n-1 \quad \text{and} \quad s_n^a < s_n^b. \tag{3.34}$$

Using this definition, the probability mass of a particular symbol sequence $\boldsymbol{s}$ can written as

$$p(\boldsymbol{s}) = P(\boldsymbol{S}=\boldsymbol{s}) = P(\boldsymbol{S}\leq\boldsymbol{s}) - P(\boldsymbol{S}<\boldsymbol{s}). \tag{3.35}$$

This expression indicates that a symbol sequence $\boldsymbol{s}$ can be represented by an interval $\mathcal{I}_N$ between two successive values of the cumulative probability mass function $P(\boldsymbol{S}\leq\boldsymbol{s})$. The corresponding mapping of a symbol sequence $\boldsymbol{s}$ to a half-open interval $\mathcal{I}_N \subset [0,1)$ is given by

$$\mathcal{I}_N(\boldsymbol{s}) = [L_N, L_N+W_N) = \big[P(\boldsymbol{S}<\boldsymbol{s}),\, P(\boldsymbol{S}\leq\boldsymbol{s})\big). \tag{3.36}$$

The interval width W_N is equal to the probability $P(\boldsymbol{S}=\boldsymbol{s})$ of the associated symbol sequence $\boldsymbol{s}$. In addition, the intervals for different realizations of the random vector $\boldsymbol{S}$ are always disjoint. This can be shown by considering two symbol sequences $\boldsymbol{s}_a$ and $\boldsymbol{s}_b$, with $\boldsymbol{s}_a < \boldsymbol{s}_b$. The lower interval boundary L_N^b of the interval $\mathcal{I}_N(\boldsymbol{s}_b)$,

$$\begin{aligned} L_N^b &= P(\boldsymbol{S}<\boldsymbol{s}_b) \\ &= P(\{\boldsymbol{S}\leq\boldsymbol{s}_a\} \cup \{\boldsymbol{s}_a < \boldsymbol{S} \leq \boldsymbol{s}_b\}) \\ &= P(\boldsymbol{S}\leq\boldsymbol{s}_a) + P(\boldsymbol{S}>\boldsymbol{s}_a,\, \boldsymbol{S}<\boldsymbol{s}_b) \\ &\geq P(\boldsymbol{S}\leq\boldsymbol{s}_a) = L_N^a + W_N^a, \end{aligned} \tag{3.37}$$

is always greater than or equal to the upper interval boundary of the half-open interval $\mathcal{I}_N(\boldsymbol{s}_a)$. Consequently, an N-symbol sequence $\boldsymbol{s}$ can be uniquely represented by any real number $v \in \mathcal{I}_N$, which can be written as binary fraction with K bits after the binary point,

$$v = \sum_{i=0}^{K-1} b_i \, 2^{i-1} = 0.b_0 b_1 \cdots b_{K-1}. \tag{3.38}$$

In order to identify the symbol sequence $\boldsymbol{s}$ we only need to transmit the bit sequence $\boldsymbol{b} = \{b_0, b_1, \cdots, b_{K-1}\}$. The Elias code for the sequence of random variables $\boldsymbol{S}$ is given by the assignment of bit sequences $\boldsymbol{b}$ to the N-symbol sequences $\boldsymbol{s}$.

For obtaining codewords that are as short as possible, we should choose the real numbers v that can be represented with the minimum amount of bits. The distance between successive binary fractions with K bits after the binary point is 2^{-K}. In order to guarantee that any binary fraction with K bits after the decimal point falls in an interval of size W_N, we need $K \geq -\log_2 W_N$ bits. Consequently, we choose

$$K = K(\boldsymbol{s}) = \lceil -\log_2 W_N \rceil = \lceil -\log_2 p(\boldsymbol{s}) \rceil, \tag{3.39}$$

where $\lceil x \rceil$ represents the smallest integer greater than or equal to x. The binary fraction v, and thus the bit sequence $\boldsymbol{b}$, is determined by

$$v = \left\lceil L_N \, 2^K \right\rceil \cdot 2^{-K}. \tag{3.40}$$

An application of the inequalities $\lceil x \rceil \geq x$ and $\lceil x \rceil < x + 1$ to (3.40) and (3.39) yields

$$L_N \leq v < L_N + 2^{-K} \leq L_N + W_N, \tag{3.41}$$

which proves that the selected binary fraction v always lies inside the interval $\mathcal{I}_N$. The Elias code obtained by choosing $K = \lceil -\log_2 W_N \rceil$ associates each N-symbol sequence $\boldsymbol{s}$ with a distinct codeword $\boldsymbol{b}$.

Iterative Coding. An important property of the Elias code is that the codewords can be iteratively constructed. For deriving the iteration rules, we consider sub-sequences $\boldsymbol{s}^{(n)} = \{s_0, s_1, \cdots, s_{n-1}\}$ that consist of the first n symbols, with $1 \leq n \leq N$, of the symbol sequence $\boldsymbol{s}$.

Each of these sub-sequences $\boldsymbol{s}^{(n)}$ can be treated in the same way as the symbol sequence $\boldsymbol{s}$. Given the interval width W_n for the sub-sequence $\boldsymbol{s}^{(n)} = \{s_0, s_1, \ldots, s_{n-1}\}$, the interval width W_{n+1} for the sub-sequence $\boldsymbol{s}^{(n+1)} = \{\boldsymbol{s}^{(n)}, s_n\}$ can be derived by

$$\begin{aligned} W_{n+1} &= P\big(\boldsymbol{S}^{(n+1)} = \boldsymbol{s}^{(n+1)}\big) \\ &= P\big(\boldsymbol{S}^{(n)} = \boldsymbol{s}^{(n)},\ S_n = s_n\big) \\ &= P\big(\boldsymbol{S}^{(n)} = \boldsymbol{s}^{(n)}\big) \cdot P\big(S_n = s_n \,\big|\, \boldsymbol{S}^{(n)} = \boldsymbol{s}^{(n)}\big) \\ &= W_n \cdot p(s_n \mid s_0, s_1, \ldots, s_{n-1}), \end{aligned} \tag{3.42}$$

with $p(s_n \mid s_0, s_1, \ldots, s_{n-1})$ being the conditional probability mass function $P(S_n = s_n \mid S_0 = s_0, S_1 = s_1, \ldots, S_{n-1} = s_{n-1})$. Similarly, the iteration rule for the lower interval border L_n is given by

$$\begin{aligned} L_{n+1} &= P\big(\boldsymbol{S}^{(n+1)} < \boldsymbol{s}^{(n+1)}\big) \\ &= P\big(\boldsymbol{S}^{(n)} < \boldsymbol{s}^{(n)}\big) + P\big(\boldsymbol{S}^{(n)} = \boldsymbol{s}^{(n)},\ S_n < s_n\big) \\ &= P\big(\boldsymbol{S}^{(n)} < \boldsymbol{s}^{(n)}\big) + P\big(\boldsymbol{S}^{(n)} = \boldsymbol{s}^{(n)}\big) \cdot P\big(S_n < s_n \,\big|\, \boldsymbol{S}^{(n)} = \boldsymbol{s}^{(n)}\big) \\ &= L_n + W_n \cdot c(s_n \mid s_0, s_1, \ldots, s_{n-1}), \end{aligned} \tag{3.43}$$

where $c(s_n \mid s_0, s_1, \ldots, s_{n-1})$ represents a cumulative probability mass function (cmf) and is given by

$$c(s_n \mid s_0, s_1, \ldots, s_{n-1}) = \sum_{\forall a \in \mathcal{A}_n :\, a < s_n} p(a \mid s_0, s_1, \ldots, s_{n-1}). \tag{3.44}$$

By setting $W_0 = 1$ and $L_0 = 0$, the iteration rules (3.42) and (3.43) can also be used for calculating the interval width and lower interval border of the first sub-sequence $\boldsymbol{s}^{(1)} = \{s_0\}$. Equation (3.43) directly implies $L_{n+1} \geq L_n$. By combining (3.43) and (3.42), we also obtain

$$\begin{aligned} L_{n+1} + W_{n+1} &= L_n + W_n \cdot P\big(S_n \leq s_n \,\big|\, \boldsymbol{S}^{(n)} = \boldsymbol{s}^{(n)}\big) \\ &= L_n + W_n - W_n \cdot P\big(S_n > s_n \,\big|\, \boldsymbol{S}^{(n)} = \boldsymbol{s}^{(n)}\big) \\ &\leq L_n + W_n. \end{aligned} \tag{3.45}$$

The interval $\mathcal{I}_{n+1}$ for a symbol sequence $\boldsymbol{s}^{(n+1)}$ is nested inside the interval $\mathcal{I}_n$ for the symbol sequence $\boldsymbol{s}^{(n)}$ that excludes the last symbol s_n.

The iteration rules have been derived for the general case of dependent and differently distributed random variables S_n. For iid processes

and Markov processes, the general conditional pmf in (3.42) and (3.44) can be replaced with the marginal pmf $p(s_n) = P(S_n = s_n)$ and the conditional pmf $p(s_n|s_{n-1}) = P(S_n = s_n|S_{n-1} = s_{n-1})$, respectively.

symbol a_k	pmf $p(a_k)$	Huffman code	cmf $c(a_k)$
a_0=‘A’	$0.50 = 2^{-2}$	00	$0.00 = 0$
a_1=‘B’	$0.25 = 2^{-2}$	01	$0.25 = 2^{-2}$
a_2=‘C’	$0.25 = 2^{-1}$	1	$0.50 = 2^{-1}$

Table 3.6 Example for an iid process with a 3-symbol alphabet.

As an example, we consider the iid process in Table 3.6. Beside the pmf $p(a)$ and cmf $c(a)$, the table also specifies a Huffman code. Suppose we intend to transmit the symbol sequence $\boldsymbol{s}$ =‘CABAC’. If we use the Huffman code, the transmitted bit sequence would be $\boldsymbol{b}$ =‘10001001’. The iterative code construction process for the Elias coding is illustrated in Table 3.7. The constructed codeword is identical to the codeword that is obtained with the Huffman code. Note that the codewords of an Elias code have only the same number of bits as the Huffman code if all probability masses are integer powers of 1/2 as in our example.

s_0=‘C’	s_1=‘A’	s_2=‘B’
$W_1 = W_0 \cdot p(\text{‘C’}) = 1 \cdot 2^{-1} = 2^{-1} = (0.1)_b$	$W_2 = W_1 \cdot p(\text{‘A’}) = 2^{-1} \cdot 2^{-2} = 2^{-3} = (0.001)_b$	$W_3 = W_2 \cdot p(\text{‘B’}) = 2^{-3} \cdot 2^{-2} = 2^{-5} = (0.00001)_b$
$L_1 = L_0 + W_0 \cdot c(\text{‘C’}) = L_0 + 1 \cdot 2^{-1} = 2^{-1} = (0.1)_b$	$L_2 = L_1 + W_1 \cdot c(\text{‘A’}) = L_1 + 2^{-1} \cdot 0 = 2^{-1} = (0.100)_b$	$L_3 = L_2 + W_2 \cdot c(\text{‘B’}) = L_2 + 2^{-3} \cdot 2^{-2} = 2^{-1} + 2^{-5} = (0.10001)_b$
s_3=‘A’	s_4=‘C’	termination
$W_4 = W_3 \cdot p(\text{‘A’}) = 2^{-5} \cdot 2^{-2} = 2^{-7} = (0.0000001)_b$	$W_5 = W_4 \cdot p(\text{‘C’}) = 2^{-7} \cdot 2^{-1} = 2^{-8} = (0.00000001)_b$	$K = \lceil -\log_2 W_5 \rceil = 8$
$L_4 = L_3 + W_3 \cdot c(\text{‘A’}) = L_3 + 2^{-5} \cdot 0 = 2^{-1} + 2^{-5} = (0.1000100)_b$	$L_5 = L_4 + W_4 \cdot c(\text{‘C’}) = L_4 + 2^{-7} \cdot 2^{-1} = 2^{-1} + 2^{-5} + 2^{-8} = (0.10001001)_b$	$v = \lceil L_5\, 2^K \rceil\, 2^{-K} = 2^{-1} + 2^{-5} + 2^{-8}$ $\boldsymbol{b} = \text{‘}\mathbf{10001001}'$

Table 3.7 Iterative code construction process for the symbol sequence ‘CABAC’. It is assumed that the symbol sequence is generated by the iid process specified in Table 3.6.

Based on the derived iteration rules, we state an iterative encoding and decoding algorithm for Elias codes. The algorithms are specified for the general case using multiple symbol alphabets and conditional pmfs and cmfs. For stationary processes, all alphabets $\mathcal{A}_n$ can be replaced by a single alphabet $\mathcal{A}$. For iid sources, Markov sources, and other simple source models, the conditional pmfs $p(s_n|s_0, \cdots, s_{n-1})$ and cmfs $c(s_n|s_0, \cdots, s_{n-1})$ can be simplified as discussed above.

Encoding algorithm:

(1) Given is a sequence $\{s_0, \cdots, s_{N-1}\}$ of N symbols.
(2) Initialization of the iterative process by $W_0 = 1$, $L_0 = 0$.
(3) For each $n = 0, 1, \cdots, N-1$, determine the interval $\mathcal{I}_{n+1}$ by

$$\begin{aligned} W_{n+1} &= W_n \cdot p(s_n|s_0, \cdots, s_{n-1}), \\ L_{n+1} &= L_n + W_n \cdot c(s_n|s_0, \cdots, s_{n-1}). \end{aligned}$$

(4) Determine the codeword length by $K = \lceil -\log_2 W_N \rceil$.
(5) Transmit the codeword $\boldsymbol{b}^{(K)}$ of K bits that represents the fractional part of $v = \lceil L_N \, 2^K \rceil \, 2^{-K}$.

Decoding algorithm:

(1) Given is the number N of symbols to be decoded and a codeword $\boldsymbol{b}^{(K)} = \{b_0, \cdots, b_{K-1}\}$ of K_N bits.
(2) Determine the interval representative v according to

$$v = \sum_{i=0}^{K-1} b_i \, 2^{-i}.$$

(3) Initialization of the iterative process by $W_0 = 1$, $L_0 = 0$.
(4) For each $n = 0, 1, \cdots, N-1$, do the following:
   (a) For each $a_i \in \mathcal{A}_n$, determine the interval $\mathcal{I}_{n+1}(a_i)$ by

$$\begin{aligned} W_{n+1}(a_i) &= W_n \cdot p(a_i|s_0, \ldots, s_{n-1}), \\ L_{n+1}(a_i) &= L_n + W_n \cdot c(a_i|s_0, \ldots, s_{n-1}). \end{aligned}$$

   (b) Select the letter $a_i \in \mathcal{A}_n$ for which $v \in \mathcal{I}_{n+1}(a_i)$, and set $s_n = a_i$, $W_{n+1} = W_{n+1}(a_i)$, $L_{n+1} = L_{n+1}(a_i)$.

Adaptive Elias Coding. Since the iterative interval refinement is the same at encoder and decoder side, Elias coding provides a simple mechanism for the adaptation to sources with unknown or nonstationary statistical properties. Conceptually, for each source symbol s_n, the pmf $p(s_n|s_0, \cdots, s_{n-1})$ can be simultaneously estimated at encoder and decoder side based on the already coded symbols s_0 to s_{n-1}. For this purpose, a source can often be modeled as a process with independent random variables or as a Markov process. For the simple model of independent random variables, the pmf $p(s_n)$ for a particular symbol s_n can be approximated by the relative frequencies of the alphabet letters inside the sequence of the preceding N_W coded symbols. The chosen interval size N_W adjusts the trade-off between a fast adaptation and an accurate probability estimation. The same approach can also be applied for high-order probability models as the Markov model. In this case, the conditional pmf is approximated by the corresponding relative conditional frequencies.

Efficiency of Elias Coding. The average codeword length per symbol for the Elias code is given by

$$\bar{\ell} = \frac{1}{N} E\{K(\boldsymbol{S})\} = \frac{1}{N} E\big\{\lceil -\log_2 p(\boldsymbol{S})\rceil\big\}\,. \tag{3.46}$$

By applying the inequalities $\lceil x \rceil \geq x$ and $\lceil x \rceil < x + 1$, we obtain

$$\begin{aligned} \frac{1}{N} E\{-\log_2 p(\boldsymbol{S})\} \leq \bar{\ell} &< \frac{1}{N} E\{1 - \log_2 p(\boldsymbol{S})\} \\ \frac{1}{N} H(S_0, \cdots, S_{N-1}) \leq \bar{\ell} &< \frac{1}{N} H(S_0, \cdots, S_{N-1}) + \frac{1}{N}. \end{aligned} \tag{3.47}$$

If the number N of coded symbols approaches infinity, the average codeword length approaches the entropy rate.

It should be noted that the Elias code is not guaranteed to be prefix free, i.e., a codeword for a particular symbol sequence may be a prefix of the codeword for any other symbol sequence. Hence, the Elias code as described above can only be used if the length of the codeword is known at the decoder side[5]. A prefix-free Elias code can be constructed

[5] In image and video coding applications, the end of a bit sequence for the symbols of a picture or slice is often given by the high-level bitstream syntax.

if the lengths of all codewords are increased by one, i.e., by choosing

$$K_N = \lceil -\log_2 W_N \rceil + 1. \tag{3.48}$$

3.4.2 Arithmetic Coding

The Elias code has several desirable properties, but it is still impractical, since the precision that is required for representing the interval widths and lower interval boundaries grows without bound for long symbol sequences. The widely-used approach of *arithmetic coding* is a variant of Elias coding that can be implemented with fixed-precision integer arithmetic.

For the following considerations, we assume that the probability masses $p(s_n|s_0, \cdots, s_{n-1})$ are given with a fixed number V of binary digit after the binary point. We will omit the conditions "$s_0, \cdots, s_{n-1}$" and represent the pmfs $p(a)$ and cmfs $c(a)$ by

$$p(a) = p_V(a) \cdot 2^{-V}, \quad c(a) = c_V(a) \cdot 2^{-V} = \sum_{a_i < a} p_V(a_i) \cdot 2^{-V}, \tag{3.49}$$

where $p_V(a)$ and $c_V(a)$ are V-bit positive integers.

The key observation for designing arithmetic coding schemes is that the Elias code remains decodable if the interval width W_{n+1} satisfies

$$0 < W_{n+1} \leq W_n \cdot p(s_n). \tag{3.50}$$

This guarantees that the interval $\mathcal{I}_{n+1}$ is always nested inside the interval $\mathcal{I}_n$. Equation (3.43) implies $L_{n+1} \geq L_n$, and by combining (3.43) with the inequality (3.50), we obtain

$$L_{n+1} + W_{n+1} \leq L_n + W_n \cdot [\, c(s_n) + p(s_n) \,] \leq L_n + W_n. \tag{3.51}$$

Hence, we can represent the interval width W_n with a fixed number of precision bits if we round it toward zero in each iteration step.

Let the interval width W_n be represented by a U-bit integer A_n and an integer $z_n \geq U$ according to

$$W_n = A_n \cdot 2^{-z_n}. \tag{3.52}$$

We restrict A_n to the range

$$2^{U-1} \leq A_n < 2^U, \tag{3.53}$$

so that the W_n is represented with a maximum precision of U bits. In order to suitably approximate $W_0 = 1$, the values of A_0 and z_0 are set equal to $2^U - 1$ and U, respectively. The interval refinement can then be specified by

$$A_{n+1} = \lfloor A_n \cdot p_V(s_n) \cdot 2^{-y_n} \rfloor, \tag{3.54}$$

$$z_{n+1} = z_n + V - y_n, \tag{3.55}$$

where y_n is a bit shift parameter with $0 \leq y_n \leq V$. These iteration rules guarantee that (3.50) is fulfilled. It should also be noted that the operation $\lfloor x \cdot 2^{-y} \rfloor$ specifies a simple right shift of the binary representation of x by y binary digits. To fulfill the constraint in (3.53), the bit shift parameter y_n has to be chosen according to

$$y_n = \lceil \log_2(A_n \cdot p_V(s_n) + 1) \rceil - U. \tag{3.56}$$

The value of y_n can be determined by a series of comparison operations.

Given the fixed-precision representation of the interval width W_n, we investigate the impact on the lower interval boundary L_n. The binary representation of the product

$$\begin{aligned} W_n \cdot c(s_n) &= A_n \cdot c_V(s_n) \cdot 2^{-(z_n+V)} \\ &= 0.\underbrace{00000\cdots0}_{z_n-U \text{ bits}}\ \underbrace{xxxxx\cdots x}_{U+V \text{ bits}}\ 00000\cdots \end{aligned} \tag{3.57}$$

consists of $z_n - U$ 0-bits after the binary point followed by $U+V$ bits representing the integer $A_n \cdot c_V(s_n)$. The bits after the binary point in the binary representation of the lower interval boundary,

$$L_n = 0.\underbrace{aaaaa\cdots a}_{\substack{z_n-c_n-U \\ \textit{settled bits}}}\ \underbrace{0111111\cdots1}_{\substack{c_n \\ \textit{outstanding bits}}}\ \underbrace{xxxxx\cdots x}_{\substack{U+V \\ \textit{active bits}}}\ \underbrace{00000\cdots}_{\textit{trailing bits}}, \tag{3.58}$$

can be classified into four categories. The *trailing bits* that follow the $(z_n + V)$-th bit after the binary point are equal to 0, but may be modified by following interval updates. The preceding $U+V$ bits are directly modified by the update $L_{n+1} = L_n + W_n\, c(s_n)$ and are referred to as *active bits*. The active bits are preceded by a sequence of zero or more

1-bits and a leading 0-bit (if present). These c_n bits are called *outstanding bits* and may be modified by a carry from the active bits. The $z_n - c_n - U$ bits after the binary point, which are referred to as *settled bits*, are not modified in any following interval update. Furthermore, these bits cannot be modified by the rounding operation that generates the final codeword, since all intervals $\mathcal{I}_{n+k}$, with $k > 0$, are nested inside the interval $\mathcal{I}_n$ and the binary representation of the interval width $W_n = A_n\, 2^{-z_n}$ also consists of $z_n - U$ 0-bits after the binary point. And since the number of bits in the final codeword,

$$K = \lceil -\log_2 W_N \rceil \geq \lceil -\log_2 W_n \rceil = z_n - \lfloor \log_2 A_n \rfloor = z_n - U + 1, \quad (3.59)$$

is always greater than or equal to the number of settled bits, the settled bits can be transmitted as soon as they have become settled. Hence, in order to represent the lower interval boundary L_n, it is sufficient to store the $U+V$ active bits and a counter for the number of 1-bits that precede the active bits.

For the decoding of a particular symbol s_n it has to be determined whether the binary fraction v in (3.40) that is represented by the transmitted codeword falls inside the interval $W_{n+1}(a_i)$ for an alphabet letter a_i. Given the described fixed-precision interval refinement, it is sufficient to compare the c_{n+1} outstanding bits and the $U+V$ active bits of the lower interval boundary L_{n+1} with the corresponding bits of the transmitted codeword and the upper interval boundary $L_{n+1} + W_{n+1}$.

It should be noted that the number of outstanding bits can become arbitrarily large. In order to force an output of bits, the encoder can insert a 0-bit if it detects a sequence of a particular number of 1-bits. The decoder can identify the additionally inserted bit and interpret it as extra carry information. This technique is for example used in the MQ-coder [72] of JPEG 2000 [41].

Efficiency of Arithmetic Coding. In comparison to Elias coding, the usage of the presented fixed precision approximation increases the codeword length for coding a symbol sequence $\boldsymbol{s} = \{s_0, s_1, \cdots, s_{N-1}\}$. Given W_N for $n = N$ in (3.52), the excess rate of arithmetic coding

over Elias coding is given by

$$\Delta\ell = \lceil -\log_2 W_N \rceil - \lceil -\log_2 p(\boldsymbol{s}) \rceil < 1 + \sum_{n=0}^{N-1} \log_2 \frac{W_n\, p(s_n)}{W_{n+1}}, \quad (3.60)$$

where we used the inequalities $\lceil x \rceil < x + 1$ and $\lceil x \rceil \geq x$ to derive the upper bound on the right side. We shall further take into account that we may have to approximate the real pmfs $p(a)$ in order to represent the probability masses as multiples of 2^{-V}. Let $q(a)$ represent an approximated pmf that is used for arithmetic coding and let $p_{\min}$ denote the minimum probability mass of the corresponding real pmf $p(a)$. The pmf approximation can always be done in a way that the difference $p(a) - q(a)$ is less than 2^{-V}, which gives

$$\frac{p(a) - q(a)}{p(a)} < \frac{2^{-V}}{p_{\min}} \qquad \Rightarrow \qquad \frac{p(a)}{q(a)} < \left(1 - \frac{2^{-V}}{p_{\min}}\right)^{-1}. \quad (3.61)$$

An application of the inequality $\lfloor x \rfloor > x - 1$ to the interval refinement (3.54) with the approximated pmf $q(a)$ yields

$$\begin{aligned} A_{n+1} &> A_n\, q(s_n)\, 2^{V-y_n} - 1 \\ W_{n+1} &> A_n\, q(s_n)\, 2^{V-y_n-z_{n+1}} - 2^{-z_{n+1}} \\ W_{n+1} &> A_n\, q(s_n)\, 2^{-z_n} - 2^{-z_{n+1}} \\ W_n\, q(s_n) - W_{n+1} &< 2^{-z_{n+1}}. \end{aligned} \quad (3.62)$$

By using the relationship $W_{n+1} \geq 2^{U-1-z_{n+1}}$, which is a direct consequence of (3.53), we obtain

$$\frac{W_n\, q(s_n)}{W_{n+1}} = 1 + \frac{W_n\, q(s_n) - W_{n+1}}{W_{n+1}} < 1 + 2^{1-U}. \quad (3.63)$$

Inserting the expressions (3.61) and (3.63) into (3.60) yields an upper bound for the increase in codeword length per symbol,

$$\Delta\bar{\ell} < \frac{1}{N} + \log_2\bigl(1 + 2^{1-U}\bigr) - \log_2\left(1 - \frac{2^{-V}}{p_{\min}}\right). \quad (3.64)$$

If we consider, for example, the coding of $N = 1000$ symbols with $U = 12$, $V = 16$, and $p_{\min} = 0.02$, the increase in codeword length in relation to Elias coding is guaranteed to be less than 0.003 bit per symbol.

Binary Arithmetic Coding. Arithmetic coding with binary symbol alphabets is referred to as *binary arithmetic coding*. It is the most popular type of arithmetic coding in image and video coding applications. The main reason for using binary arithmetic coding is its reduced complexity. It is particularly advantageous for adaptive coding, since the rather complex estimation of M-ary pmfs can be replaced by the simpler estimation of binary pmfs. Well-known examples of efficient binary arithmetic coding schemes that are used in image and video coding are the MQ-coder [72] in the picture coding standard JPEG 2000 [41] and the M-coder [55] in the video coding standard H.264/AVC [36].

In general, a symbol sequence $\boldsymbol{s} = \{s_0, s_1, \cdots, s_{N-1}\}$ has to be first converted into a sequence $\boldsymbol{c} = \{c_0, c_1, \cdots, c_{B-1}\}$ of binary symbols, before binary arithmetic coding can be applied. This conversion process is often referred to as *binarization* and the elements of the resulting binary sequences $\boldsymbol{c}$ are also called *bins*. The number B of bins in a sequence $\boldsymbol{c}$ can depend on the actual source symbol sequence $\boldsymbol{s}$. Hence, the bin sequences $\boldsymbol{c}$ can be interpreted as realizations of a variable-length sequence of binary random variables $\boldsymbol{C} = \{C_0, C_1, \cdots, C_{B-1}\}$.

Conceptually, the binarization mapping $\boldsymbol{S} \rightarrow \boldsymbol{C}$ represents a lossless coding step and any lossless source code could be applied for this purpose. It is, however, only important that the used lossless source code is uniquely decodable. The average codeword length that is achieved by the binarization mapping does not have any impact on the efficiency of binary arithmetic coding, since the block entropy for the sequence of random variables $\boldsymbol{S} = \{S_0, S_1, \cdots, S_{N-1}\}$,

$$H(\boldsymbol{S}) = E\{-\log_2 p(\boldsymbol{S})\} = E\{-\log_2 p(\boldsymbol{C})\} = H(\boldsymbol{C}),$$

is equal to entropy of the variable-length binary random vector $\boldsymbol{C} = \{C_0, C_1, \cdots, C_{B-1}\}$. The actual compression is achieved by the arithmetic coding. The above result also shows that binary arithmetic coding can provide the same coding efficiency as M-ary arithmetic coding, if the influence of the finite precision arithmetic is negligible.

In practice, the binarization is usually done with very simple prefix codes for the random variables S_n. As we assume that the order of different random variables is known to both, encoder and decoder, different prefix codes can be used for each random variable without

impacting unique decodability. A typical example for a binarization mapping, which is called truncated unary binarization, is illustrated in Table 3.8.

S_n	number of bins B	C_0	C_1	C_2	$\cdots$	C_{M-2}	C_{M-1}
a_0	1	1					
a_1	2	0	1				
a_2	3	0	0	1			
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\ddots$	$\ddots$		
a_{M-3}	$M-3$	0	0	0	$\ddots$	1	
a_{M-2}	$M-2$	0	0	0	$\cdots$	0	1
a_{M-1}	$M-2$	0	0	0	$\cdots$	0	0

Table 3.8 Mapping of a random variable S_n with an M-ary alphabet onto a variable-length binary random vector $\boldsymbol{C} = \{C_0, C_1, \ldots, C_{B-1}\}$ using truncated unary binarization.

The binary pmfs for the random variables C_i can be directly derived from the pmfs of the random variables S_n. For the example in Table 3.8, the binary pmf $\{P(C_i{=}0), 1 - P(C_i{=}0)\}$ for a random variable C_i is given by

$$P(C_i{=}0) = \frac{P(S_n > a_i \,|\, S_0{=}s_0,\, S_1{=}s_1,\, \ldots,\, S_{n-1}{=}s_{n-1})}{P(S_n \geq a_i \,|\, S_0{=}s_0,\, S_1{=}s_1,\, \ldots,\, S_{n-1}{=}s_{n-1})}, \qquad (3.65)$$

where we omitted the condition for the binary pmf. For coding nonstationary sources, it is usually preferable to directly estimate the marginal or conditional pmf's for the binary random variables instead of the pmf's for the source signal.

3.5 Probability Interval Partitioning Entropy Coding

For a some applications, arithmetic coding is still considered as too complex. As a less-complex alternative, a lossless coding scheme called probability interval partitioning entropy (PIPE) coding has been recently proposed [54]. It combines concepts from binary arithmetic coding and Huffman coding for variable-length vectors with a quantization of the binary probability interval.

A block diagram of the PIPE coding structure is shown in Fig. 3.4. It is assumed that the input symbol sequences $\boldsymbol{s} = \{s_0, s_1, \cdots, s_{N-1}\}$

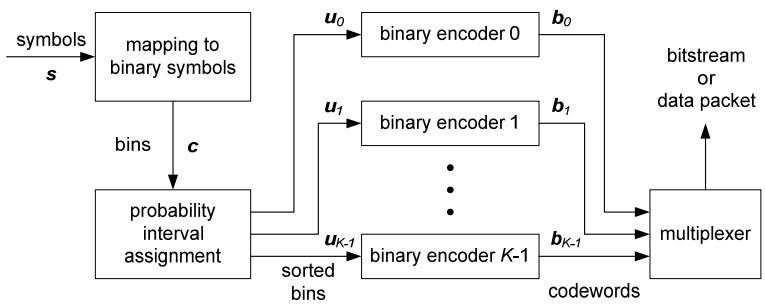


Fig. 3.4 Overview of the PIPE coding structure.

represent realizations of a sequence $\boldsymbol{S} = \{S_0, S_1, \cdots, S_{N-1}\}$ of random variables. Each random variable can be characterized by a distinct alphabet $\mathcal{A}_n$. The number N of source symbols is assumed to be known to encoder and decoder. Similarly as for binary arithmetic coding, a symbol sequence $\boldsymbol{s} = \{s_0, s_1, \cdots, s_{N-1}\}$ is first converted into a sequence $\boldsymbol{c} = \{c_0, c_1, \cdots, c_{B-1}\}$ of B binary symbols (bins). Each bin c_i can be considered as a realization of a corresponding random variable C_i and is associated with a pmf. The binary pmf is given by the probability $P(C_i = 0)$, which is known to encoder and decoder. Note that the conditional dependencies have been omitted in order to simplify the description.

The key observation for designing a low-complexity alternative to binary arithmetic coding is that an appropriate quantization of the binary probability interval has only a minor impact on the coding efficiency. This is employed by partitioning the binary probability interval into a small number U of half-open intervals $\mathcal{I}_k = (p_k, p_{k+1}]$, with $0 \leq k < U$. Each bin c_i is assigned to the interval $\mathcal{I}_k$ for which $p_k < P(C_i = 0) \leq p_{k+1}$. As a result, the bin sequence $\boldsymbol{c}$ is decomposed into U bin sequences $\boldsymbol{u}_k = \{u_0^k, u_1^k, \ldots\}$, with $0 \leq k < U$. For the purpose of coding, each of the bin sequences $\boldsymbol{u}_k$ can be treated as a realization of a binary iid process with a pmf $\{p_{\mathcal{I}_k}, 1 - p_{\mathcal{I}_k}\}$, where $p_{\mathcal{I}_k}$ denotes a representative probability for an interval $\mathcal{I}_k$, and can be efficiently coded with a V2V code as described in sec. 3.3. The resulting U codeword sequences $\boldsymbol{b}_k$ are finally multiplexed in order to produce a data packet for the symbol sequence $\boldsymbol{s}$.

Given the U probability intervals $\mathcal{I}_k = (p_k, p_{k+1}]$ and corresponding V2V codes, the PIPE coding process can be summarized as follows:

(1) *Binarization*: The sequence $\boldsymbol{s}$ of N input symbols is converted into a sequence $\boldsymbol{c}$ of B bins. Each bin c_i is characterized by a probability $P(C_i = 0)$.

(2) *Decomposition*: The bin sequence $\boldsymbol{c}$ is decomposed into U sub-sequences. A sub-sequence $\boldsymbol{u}_k$ contains the bins c_i with $P(C_i = 0) \in \mathcal{I}_k$ in the same order as in the bin sequence $\boldsymbol{c}$.

(3) *Binary Coding*: Each sub-sequence of bins $\boldsymbol{u}_k$ is coded using a distinct V2V code resulting in U codeword sequences $\boldsymbol{b}_k$.

(4) *Multiplexing*: The data packet is produced by multiplexing the U codeword sequences $\boldsymbol{b}_k$.

Binarization. The binarization process is the same as for binary arithmetic coding described in sec. 3.4. Typically, each symbol s_n of the input symbol sequence $\boldsymbol{s} = \{s_0, s_1, \cdots, s_{N-1}\}$ is converted into a sequence $\boldsymbol{c}_n$ of a variable number of bins using a simple prefix code and these bin sequences $\boldsymbol{c}_n$ are concatenated to produce the bin sequence $\boldsymbol{c}$ that uniquely represents the input symbol sequence $\boldsymbol{s}$. Here, a distinct prefix code can be used for each random variable S_n. Given the prefix codes, the conditional binary pmfs

$$p(c_i|c_0, \cdots, c_{i-1}) = P(C_i = c_i \,|\, C_0 = c_0, \cdots, C_{i-1} = c_{i-1})$$

can be directly derived based on the conditional pmfs for the random variables S_n. The binary pmfs can either be fixed or they can be simultaneously estimated at encoder and decoder side[6]. In order to simplify the following description, we omit the conditional dependencies and specify the binary pmf for the i-th bin by the probability $P(C_i = 0)$.

For the purpose of binary coding, it is preferable to use bin sequences $\boldsymbol{c}$ for which all probabilities $P(C_i = 0)$ are less than or equal to 0.5. This property can be ensured by inverting a bin value c_i if the associated probability $P(C_i = 0)$ is greater than 0.5. The inverse operation can be done at the decoder side, so that the unique decodability

[6] It is also possible to estimate the symbol pmfs, but usually a more suitable probability modeling is obtained by directly estimated the binary pmfs.

of a symbol sequence $\boldsymbol{s}$ from the associated bin sequence $\boldsymbol{c}$ is not influenced. For PIPE coding, we assume that this additional operation is done during the binarization and that all bins c_i of a bin sequence $\boldsymbol{c}$ are associated with probabilities $P(C_i=0) \leq 0.5$.

$C_i(S_n)$	$C_0(S_n)$	$C_1(S_n)$
$P(C_i(S_n)=0 \mid S_{n-1}=a_0)$	0.10	0.50
$P(C_i(S_n)=0 \mid S_{n-1}=a_1)$	0.15	1/17
$P(C_i(S_n)=0 \mid S_{n-1}=a_2)$	0.25	0.20

Table 3.9 Bin probabilities for the binarization of the stationary Markov source that is specified in Table 3.2. The truncated unary binarization as specified in Table 3.8 is applied, including bin inversions for probabilities $P(C_i=0) > 0.5$.

As an example, we consider the binarization for the stationary Markov source in specified in Table 3.2. If the truncated unary binarization given in Table 3.8 is used and all bins with probabilities $P(C_i=0)$ greater than 0.5 are inverted, we obtain the bin probabilities given in Table 3.9. $C_i(S_n)$ denotes the random variable that corresponds to the i-th bin inside the bin sequences for the random variable S_n.

Probability Interval Partitioning. The half-open probability interval $(0, 0.5]$, which includes all possible bin probabilities $P(C_i=0)$, is partitioned into U intervals $\mathcal{I}_k = (p_k, p_{k+1}]$. This set of intervals is characterized by $U-1$ interval borders p_k with $k = 1, \cdots, U-1$. Without loss of generality, we assume $p_k < p_{k+1}$. The outer interval borders are fixed and given by $p_0 = 0$ and $p_U = 0.5$. Given the interval boundaries, the sequence of bins $\boldsymbol{c}$ is decomposed into U separate bin sequences $\boldsymbol{u}_k = (u_0^k, u_1^k, \cdots)$, where each bin sequence $\boldsymbol{u}_k$ contains the bins c_i with $P(C_i=0) \in \mathcal{I}_k$. Each bin sequence $\boldsymbol{u}_k$ is coded with a binary coder that is optimized for a representative probability $p_{\mathcal{I}_k}$ for the interval $\mathcal{I}_k$.

For analyzing the impact of the probability interval partitioning, we assume that we can design a lossless code for binary iid processes that achieves the entropy limit. The average codeword length $\ell_b(p, p_{\mathcal{I}_k})$ for coding a bin c_i with the probability $p = P(C_i=0)$ using an optimal code for the representative probability $p_{\mathcal{I}_k}$ is given by

$$\ell_b(p, p_{\mathcal{I}_k}) = -p \log_2 p_{\mathcal{I}_k} - (1-p) \log_2(1 - p_{\mathcal{I}_k}). \tag{3.66}$$

When we further assume that the relative frequencies of the bin probabilities p inside a bin sequence $\boldsymbol{c}$ are given by the pdf $f(p)$, the average codeword length per bin $\bar{\ell}_b$ for a given set of U intervals $\mathcal{I}_k$ with representative probabilities $p_{\mathcal{I}_k}$ can then be written as

$$\bar{\ell}_b = \sum_{k=0}^{K-1} \left(\int_{p_k}^{p_{k+1}} \ell_b(p, p_{\mathcal{I}_k}) \, f(p) \, \mathrm{d}p. \right). \tag{3.67}$$

Minimization with respect to the interval boundaries p_k and representative probabilities $p_{\mathcal{I}_k}$ yields the equation system,

$$p^*_{\mathcal{I}_k} = \frac{\int_{p_k}^{p_{k+1}} p \, f(p) \, \mathrm{d}p}{\int_{p_k}^{p_{k+1}} f(p) \, \mathrm{d}p}, \tag{3.68}$$

$$p^*_k = p \quad \text{with} \quad \ell_b(p, p_{\mathcal{I}_{k-1}}) = \ell_b(p, p_{\mathcal{I}_k}). \tag{3.69}$$

Given the pdf $f(p)$ and the number of intervals U, the interval partitioning can be derived by an iterative algorithm that alternately updates the interval borders p_k and interval representatives $p_{\mathcal{I}_k}$. As an example, Fig. 3.5 shows the probability interval partitioning for a uniform distribution $f(p)$ of the bin probabilities and $U = 4$ intervals. As can be seen, the probability interval partitioning leads to a piecewise linear approximation $\ell_b(p, p_{\mathcal{I}_k})|_{\mathcal{I}_k}$ of the binary entropy function $H(p)$.

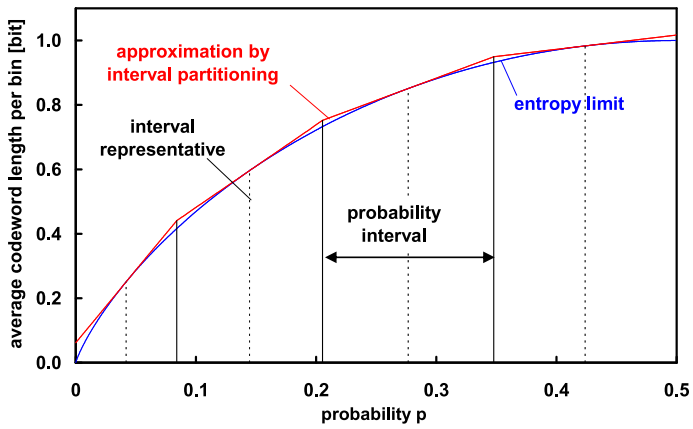


Fig. 3.5 Example for the partitioning of the probability interval $(0, 0.5]$ into 4 intervals assuming a uniform distribution of the bin probabilities $p = P(C_i = 0)$.

The increase of the average codeword length per bin is given by

$$\bar{\varrho} = \bar{\ell}_b \,/\, \left(\int_0^{0.5} H(p)\, f(p)\, \mathrm{d}p \right) - 1. \tag{3.70}$$

Table 3.10 lists the increases in average codeword length per bin for a uniform and a linear increasing ($f(p) = 8p$) distribution of the bin probabilities for selected numbers U of intervals.

U	1	2	4	8	12	16
$\bar{\varrho}_{\text{uni}}$ [%]	12.47	3.67	1.01	0.27	0.12	0.07
$\bar{\varrho}_{\text{lin}}$ [%]	5.68	1.77	0.50	0.14	0.06	0.04

Table 3.10 Increase in average codeword length per bin for a uniform and a linear increasing distribution $f(p)$ of bin probabilities and various numbers of probability intervals.

We now consider the probability interval partitioning for the Markov source specified in Table 3.2. As shown in Table 3.9, the binarization described above led to 6 different bin probabilities. For the truncated unary binarization of a Markov source, the relative frequency $h(p_{ij})$ that a bin with probability $p_{ij} = P(C_i(S_n)|S_{n-1}=a_j)$ occurs inside the bin sequence $\boldsymbol{c}$ is equal to

$$h(p_{ij}) = \frac{p(a_j) \sum_{k=i}^{M-1} p(a_k|a_j)}{\sum_{m=0}^{M-2} \sum_{k=m}^{M-1} p(a_k)}. \tag{3.71}$$

The distribution of the bin probabilities is given by

$$\begin{aligned} f(p) \;=\; & 0.1533 \cdot \delta(p-1/17) + 0.4754 \cdot \delta(p-0.1) + 0.1803 \cdot \delta(p-0.15) + \\ & 0.0615 \cdot \delta(p-0.2) + 0.0820 \cdot \delta(p-0.25) + 0.0475 \cdot \delta(p-0.5), \end{aligned}$$

where δ represents the Direct delta function. An optimal partitioning of the probability interval $(0, 0.5]$ into 3 intervals for this source is shown in Table 3.11. The increase in average codeword length per bin for this example is approximately 0.85%.

Binary Coding. For the purpose of binary coding, a bin sequence $\boldsymbol{u}_k$ for the probability interval $\mathcal{I}_k$ can be treated as a realization of a binary iid process with a pmf $\{p_{\mathcal{I}_k}, 1 - p_{\mathcal{I}_k}\}$. The statistical dependencies between the bins have already been exploited by associating each bin c_i

interval $\mathcal{I}_k = (p_k, p_{k+1}]$	representative $p_{\mathcal{I}_k}$
$\mathcal{I}_0 = (0, 0.1326]$	0.09
$\mathcal{I}_1 = (0.1326, 0.3294]$	0.1848
$\mathcal{I}_2 = (0.3294, 0.5]$	0.5000

Table 3.11 Optimal partitioning of the probability interval $(0, 0.5]$ into 3 intervals for a truncated unary binarization of the Markov source specified in Table 3.2.

with a probability $P(C_i = 0)$ that depends on previously coded bins or symbols according to the employed probability modeling. The V2V codes described in sec. 3.3 are simple but very efficient lossless source codes for binary iid processes $\boldsymbol{U}_k = \{U_n^k\}$. Using these codes, a variable number of bins is mapped to a variable-length codeword. By considering a sufficiently large number of tables entries, these codes can achieve an average codeword length close to the entropy rate $\bar{H}(\boldsymbol{U}_k) = H(U_n^k)$.

$p_{\mathcal{I}_0} = 0.09$		$p_{\mathcal{I}_1} = 0.1848$		$p_{\mathcal{I}_2} = 0.5$	
$\bar{\ell}_0 = 0.4394,\ \varrho_0 = 0.69\%$		$\bar{\ell}_1 = 0.6934,\ \varrho_1 = 0.42\%$		$\bar{\ell}_2 = 1,\ \varrho_2 = 0\%$	
bin sequence	codeword	bin sequence	codeword	bin sequence	codeword
'1111111'	'1'	'111'	'1'	'1'	'1'
'0'	'011'	'110'	'001'	'0'	'0'
'10'	'0000'	'011'	'010'		
'110'	'0001'	'1011'	'011'		
'1110'	'0010'	'00'	'00000'		
'11110'	'0011'	'100'	'00001'		
'111110'	'0100'	'010'	'00010'		
'1111110'	'0101'	'1010'	'00011'		

Table 3.12 Optimal V2V codes with up to eight codeword entries for the interval representatives $p_{\mathcal{I}_k}$ of the probability interval partitioning specified in Table 3.11.

As an example, Table 3.12 shows V2V codes for the interval representatives $p_{\mathcal{I}_k}$ of the probability interval partitioning given in Table 3.11. These codes achieve the minimum average codeword length per bin among all V2V codes with up to 8 codewords. The table additionally lists the average codeword lengths per bin $\bar{\ell}_k$ and the corresponding redundancies $\varrho_k = \big(\bar{\ell}_k - \bar{H}(\boldsymbol{U}_k)\big)/\bar{H}(\boldsymbol{U}_k)$. The code redundancies could be further decreased if V2V codes with more than 8 codewords are considered. When we assume that the number of N of symbols approaches infinity, the average codeword length per symbol for the

applied truncated unary binarization is given by

$$\bar{\ell} = \left(\sum_{k=0}^{U-1} \bar{\ell}_k \int_{p_k}^{p_{k+1}} f(p)\,\mathrm{d}p \right) \cdot \left(\sum_{m=0}^{M-2} \sum_{k=m}^{M-1} p(a_k) \right), \tag{3.72}$$

where the first term represents the average codeword length per bin for the bin sequence $\boldsymbol{c}$ and the second term is the bin-to-symbol ratio. For our simple example, the average codeword length for the PIPE coding is $\bar{\ell} = 0.7432$ bit per symbol. It is only 1.37% larger than the entropy rate and significantly smaller than the average codeword length for the scalar, conditional, and block Huffman codes that we have developed in sec. 3.2 and sec. 3.3.

In general, the average codeword length per symbol can be further decreased if the V2V codes and the probability interval partitioning are jointly optimized. This can be achieved by an iterative algorithm that alternately optimizes the interval representatives $p_{\mathcal{I}_k}$, the V2V codes for the interval representatives, and the interval boundaries p_k. Each codeword entry m of a binary V2V code $\mathcal{C}_k$ is characterized by the number x_m of 0-bins, the number y_m of 1-bins, and the length ℓ_m of the codeword. As can be concluded from the description of V2V codes in sec. 3.3, the average codeword length for coding a bin c_i with a probability $p = P(C_i = 0)$ using a V2V code $\mathcal{C}_k$ is given by

$$\bar{\ell}_b(p, \mathcal{C}_k) = \frac{\sum_{m=0}^{V-1} p^{x_m} (1-p)^{y_m} \ \ell_m}{\sum_{m=0}^{V-1} p^{x_m} (1-p)^{y_m} \ (x_m + y_m)}, \tag{3.73}$$

where V denotes the number of codeword entries. Hence, an optimal interval border p_k is given by the intersection point of the functions $\bar{\ell}_b(p, \mathcal{C}_{k-1})$ and $\bar{\ell}_b(p, \mathcal{C}_k)$ for the V2V codes of the neighboring intervals.

As an example, we jointly derived the partitioning into $U = 12$ probability intervals and corresponding V2V codes with up to 65 codeword entries for a uniform distribution of bin probabilities. Fig. 3.6 shows the difference between the average codeword length per bin and the binary entropy function $H(p)$ for this design and a theoretically optimal probability interval partitioning assuming optimal binary codes with $\bar{\ell}_k = H(p_{\mathcal{I}_k})$. The overall redundancy with respect to the entropy limit is 0.24% for the jointly optimized design and 0.12% for the probability interval partitioning assuming optimal binary codes.

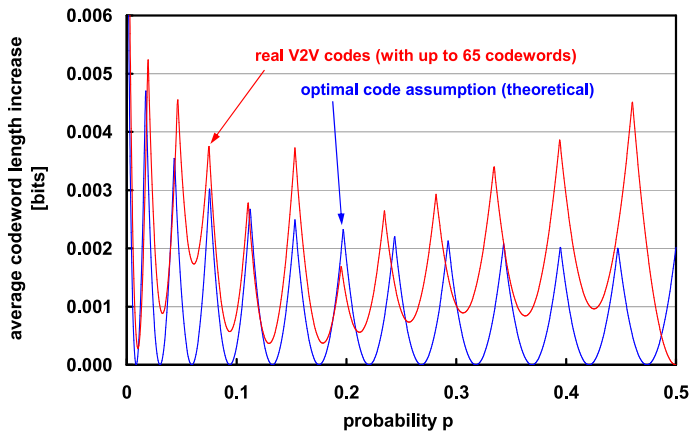


Fig. 3.6 Difference between the average codeword length and the binary entropy function $H(p)$ for a probability interval partitioning into $U = 12$ intervals assuming optimal binary codes and a real design with V2V codes of up to 65 codeword entries. The distribution of bin probabilities is assumed to be uniform.

Multiplexing. The U codeword sequences $\boldsymbol{b}_k$ that are generated by the different binary encoders for a set of source symbols (e.g., a slice of a video picture) can be written to different partitions of a data packet. This enables a parallelization of the bin encoding and decoding process. At the encoder side, each sub-sequence $\boldsymbol{u}_k$ is written to a different buffer and the actual binary encoding can be done in parallel. At the decoder side, the U codeword sequences $\boldsymbol{b}_k$ can be decoded in parallel and the resulting bin sequences $\boldsymbol{u}_k$ can be stored in separate bin buffers. The remaining entropy decoding process can then be designed in a way that it simply reads bins from the corresponding U bin buffers.

The separate transmission of the codeword streams requires the signaling of partitioning information. Furthermore, parallelized entropy coding is often not required for small data packets. In such a case, the codewords of the U codeword sequences can be interleaved without any rate overhead. The decoder can simply read a new codeword from the bitstream if a new bin is requested by the decoding process and all bins of the previously read codeword for the corresponding interval $\mathcal{I}_k$ have been used. At the encoder side, it has to be ensured that the codewords are written in the same order in which they are read at the decoder side. This can be efficiently realized by introducing a codeword buffer.

Unique Decodability. For PIPE coding, the concept of unique decodability has to be extended. Since the binarization is done using prefix codes, it is always invertible[7]. However, the resulting sequence of bins $\boldsymbol{c}$ is partitioned into U sub-sequences $\boldsymbol{u}_k$

$$\{\boldsymbol{u}_0, \ldots, \boldsymbol{u}_{U-1}\} = \gamma_p(\boldsymbol{b}), \tag{3.74}$$

and each of these sub-sequences $\boldsymbol{u}_k$ is separately coded. The bin sequence $\boldsymbol{c}$ is uniquely decodable, if each sub-sequence of bins $\boldsymbol{u}_k$ is uniquely decodable and the partitioning rule γ_p is known to the decoder. The partitioning rule γ_p is given by the probability interval partitioning $\{\mathcal{I}_k\}$ and the probabilities $P(C_i = 0)$ that are associated with the coding bins c_i. Hence, the probability interval partitioning $\{\mathcal{I}_k\}$ has to be known at the decoder side and the probability $P(C_i = 0)$ for each bin c_i has to be derived in the same way at encoder and decoder side.

3.6 Comparison of Lossless Coding Techniques

In the preceding sections, we presented different lossless coding techniques. We now compare these techniques with respect to their coding efficiency for the stationary Markov source specified in Table 3.2 and different message sizes L. In Fig. 3.7, the average codeword lengths per symbol for the different lossless source codes are plotted over the number L of coded symbols. For each number of coded symbols, the shown average codeword lengths were calculated as mean values over a set of one million different realizations of the example Markov source and can be considered as accurate approximations of the expected average codeword lengths per symbol. For comparison, Fig. 3.7 also shows the entropy rate and the instantaneous entropy rate, which is given by

$$\bar{H}_{\text{inst}}(\boldsymbol{S}, L) = \frac{1}{L} H(S_0, S_1, \ldots, S_{L-1}) \tag{3.75}$$

and represents the greatest lower bound for the average codeword length per symbol when a message of L symbols is coded.

[7] The additionally introduced bin inversion depending on the associated probabilities $P(C_i = 0)$ is invertible, if the probabilities $P(C_i = 0)$ are derived in the same way at encoder and decoder side as stated below.

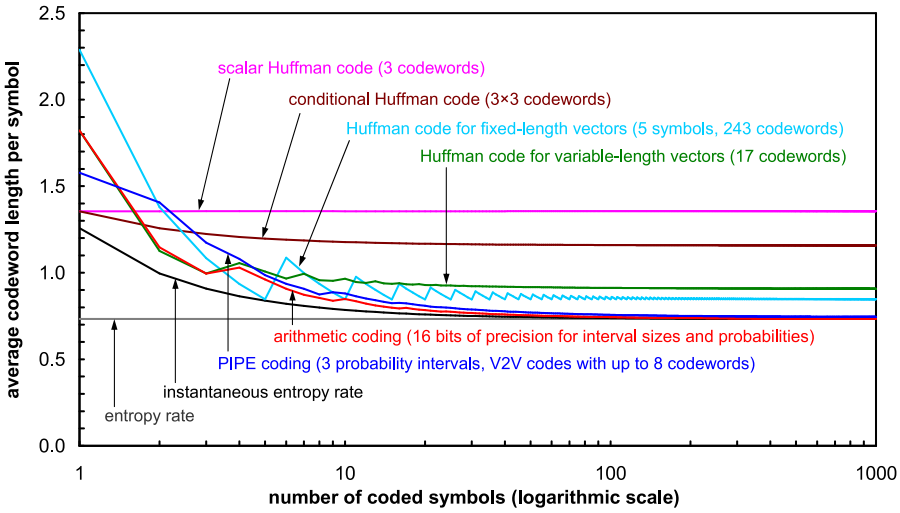


Fig. 3.7 Comparison of lossless coding techniques for the stationary Markov source specified in Table 3.2 and different numbers L of coded symbols.

For $L = 1$ and $L = 5$, the scalar Huffman code and the Huffman code for blocks of 5 symbols achieve the minimum average codeword length, respectively, which confirms that Huffman codes are optimal codes for a given set of letters or letter sequences with a fixed pmf. But if more than 10 symbols are coded, all investigated Huffman codes have a lower coding efficiency than arithmetic and PIPE coding. For large numbers of coded symbols, the average codeword length for arithmetic coding approaches the entropy rate. The average codeword length for PIPE coding is only a little bit larger; the difference to arithmetic coding could be further reduced by increasing the number of probability intervals and the number of codewords for the V2V tables.

3.7 Adaptive Coding

The design of Huffman codes and the coding process for arithmetic codes and PIPE codes require that the statistical properties of a source, i.e., the marginal pmf or the joint or conditional pmf's of up to a certain order, are known. Furthermore, the local statistical properties of real data such as image and video signals usually change with time. The

average codeword length can be often decreased if a lossless code is flexible and can be adapted to the local statistical properties of a source. The approaches for adaptive coding are classified into approaches with *forward adaptation* and approaches with *backward adaptation*. The basic coding structure for these methods is illustrated in Fig. 3.8.

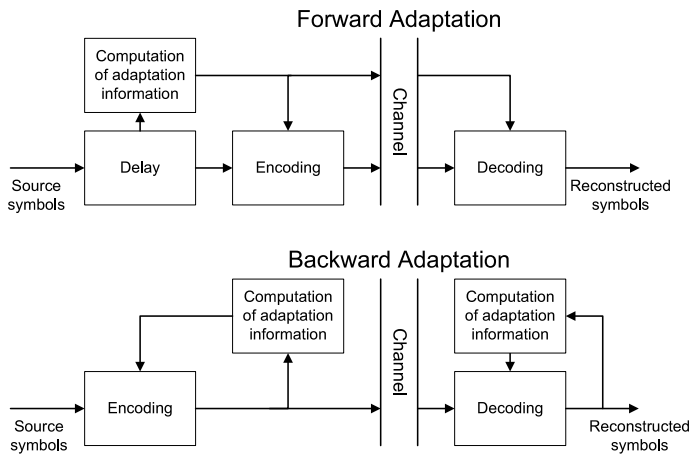


Fig. 3.8 Adaptive lossless coding with forward and backward adaptation.

In adaptive coding methods with forward adaptation, the statistical properties of a block of successive samples are analyzed in the encoder and an adaptation signal is included in the bitstream. This adaptation signal can be for example a Huffman code table, one or more pmf's, or an index into a predefined list of Huffman codes or pmf's. The decoder adjusts the used code for the block of samples according to the transmitted information. Disadvantages of this approach are that the required side information increase the transmission rate and that forward adaptation introduces a delay.

Methods with backward adaptation estimate the local statistical properties based on already coded symbols simultaneously at encoder and decoder side. As mentioned in sec. 3.2, the adaptation of Huffman codes is a quite complex task, so that backward adaptive VLC coding is rarely used in practice. But for arithmetic coding, in particular binary arithmetic coding, and PIPE coding, the backward adaptive estimation of pmf's can be easily integrated in the coding process. Backward

adaptive coding methods do not introduce a delay and do not require the transmission of any side information. However, they are not robust against transmission errors. For this reason, backward adaptation is usually only used inside a transmission packet. It is also possible to combine backward and forward adaptation. As an example, the arithmetic coding design in H.264/AVC [36] supports the transmission of a parameter inside a data packet that specifies one of three initial sets of pmf's, which are then adapted based on the actually coded symbols.

3.8 Summary of Lossless Source Coding

We have introduced the concept of uniquely decodable codes and investigated the design of prefix codes. Prefix codes provide the useful property of instantaneous decodability and it is possible to achieve an average codeword length that is not larger than the average codeword length for any other uniquely decodable code. The measures of entropy and block entropy have been derived as lower bounds for the average codeword length for coding a single symbol and a block of symbols, respectively. A lower bound for the average codeword length per symbol for any lossless source coding technique is the entropy rate.

Huffman codes have been introduced as optimal codes that assign a separate codeword to a given set of letters or letter sequences with a fixed pmf. However, for sources with memory, an average codeword length close to the entropy rate can only be achieved if a large number of symbols is coded jointly, which requires large codeword tables and is not feasible in practical coding systems. Furthermore, the adaptation of Huffman codes to time-varying statistical properties is typically considered as too complex for video coding applications, which often have real-time requirements.

Arithmetic coding represents a fixed-precision variant of Elias coding and can be considered as a universal lossless coding method. It does not require the storage of a codeword table. The arithmetic code for a symbol sequence is iteratively constructed by successively refining a cumulative probability interval, which requires a fixed number of arithmetic operations per coded symbol. Arithmetic coding can be elegantly combined with backward adaptation to the local statistical behavior of

the input source. For the coding of long symbol sequences, the average codeword length per symbol approaches the entropy rate.

As an alternative to arithmetic coding, we presented the probability interval partitioning entropy (PIPE) coding. The input symbols are binarized using simple prefix codes and the resulting sequence of binary symbols is partitioned into a small number of bin sequences, which are then coded using simple binary V2V codes. PIPE coding provides the same simple mechanism for probability modeling and backward adaptation as arithmetic coding. However, the complexity is reduced in comparison to arithmetic coding and PIPE coding provides the possibility to parallelize the encoding and decoding process. For long symbol sequences, the average codeword length per symbol is similar to that of arithmetic coding.

It should be noted that there are various other approaches to lossless coding including Lempel-Ziv coding [79], Tunstall coding [73, 66], or Burrows-Wheeler coding [7]. These methods are not considered in this text, since they are not used in the video coding area.

4

Rate Distortion Theory

In lossy coding, the reconstructed signal is not identical to the source signal, but represents only an approximation of it. A measure of the deviation between the approximation and the original signal is referred to as distortion. Rate distortion theory addresses the problem of determining the minimum average number of bits per sample that is required for representing a given source without exceeding a given distortion. The greatest lower bound for the average number of bits is referred to as the rate distortion function and represents a fundamental bound on the performance of lossy source coding algorithms, similarly as the entropy rate represents a fundamental bound for lossless source coding. For deriving the results of rate distortion theory, no particular coding technique is assumed. The applicability of rate distortion theory includes discrete and continuous random processes.

In this chapter, we give an introduction to rate distortion theory and derive rate distortion bounds for some important model processes. We will use these results in the following chapters for evaluating the performance of different lossy coding techniques. For further details, the reader is referred to the comprehensive treatments of the subject in [22, 4] and the overview in [11].

4.1 The Operational Rate Distortion Function

A lossy source coding system as illustrated in Fig. 4.1 consists of an encoder and a decoder. Given a sequence of source symbols $\boldsymbol{s}$, the encoder generates a sequence of codewords $\boldsymbol{b}$. The decoder converts the sequence of codewords $\boldsymbol{b}$ into a sequence of reconstructed symbols $\boldsymbol{s}'$.

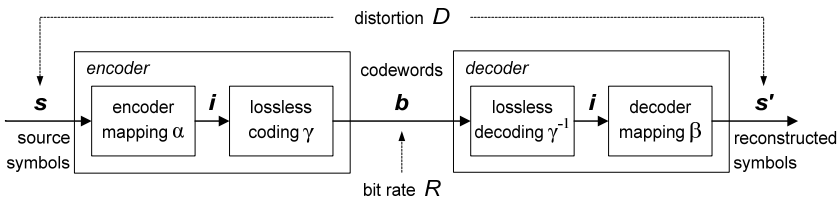


Fig. 4.1 Block diagram for a typical lossy source coding system.

The encoder operation can be decomposed into an irreversible encoder mapping α, which maps a sequence of input samples $\boldsymbol{s}$ onto a sequence of indexes $\boldsymbol{i}$, and a lossless mapping γ, which converts the sequence of indexes $\boldsymbol{i}$ into a sequence of codewords $\boldsymbol{b}$. The encoder mapping α can represent any deterministic mapping that produces a sequence of indexes $\boldsymbol{i}$ of a countable alphabet. This includes the methods of scalar quantization, vector quantization, predictive coding, and transform coding, which will be discussed in the following chapters. The lossless mapping γ can represent any lossless source coding technique, including the techniques that we discussed in chapter 3. The decoder operation consists of a lossless mapping γ^{-1}, which represents the inverse of the lossless mapping γ and converts the sequence of codewords $\boldsymbol{b}$ into the sequence of indexes $\boldsymbol{i}$, and a deterministic decoder mapping β, which maps the sequence of indexes $\boldsymbol{i}$ to a sequence of reconstructed symbols $\boldsymbol{s}'$. A lossy source coding system Q is characterized by the mappings α, β, and γ. The triple $Q = (\alpha, \beta, \gamma)$ is also referred to as *source code* or simply as *code* throughout this text.

A simple example for a source code is an N-dimensional block code $Q_N = \{\alpha_N, \beta_N, \gamma_N\}$, by which blocks of N consecutive input samples are independently coded. Each block of input samples $\boldsymbol{s}^{(N)} = \{s_0, \cdots, s_{N-1}\}$ is mapped to a vector of K quantization indexes $\boldsymbol{i}^{(K)} = \alpha_N(\boldsymbol{s}^{(N)})$ using a deterministic mapping α_N and the

resulting vector of indexes $\boldsymbol{i}$ is converted into a variable-length bit sequence $\boldsymbol{b}^{(\ell)} = \gamma_N(\boldsymbol{i}^{(K)})$. At the decoder side, the recovered vector $\boldsymbol{i}^{(K)} = \gamma_N^{-1}(\boldsymbol{b}^{(\ell)})$ of indexes is mapped to a block $\boldsymbol{s'}^{(N)} = \beta_N(\boldsymbol{i}^{(K)})$ of N reconstructed samples using the deterministic decoder mapping β_N.

In the following, we will use the notations α_N, β_N, and γ_N also for representing the encoder, decoder, and lossless mappings for the first N samples of an input sequence, independently of whether the source code Q represents an N-dimensional block code.

4.1.1 Distortion

For continuous random processes, the encoder mapping α cannot be invertible, since real numbers cannot be represented by indexes of a countable alphabet and they cannot be losslessly described by a finite number of bits. Consequently, the reproduced symbol sequence $\boldsymbol{s}'$ is not the same as the original symbol sequence $\boldsymbol{s}$. In general, if the decoder mapping β is not the inverse of the encoder mapping α, the reconstructed symbols are only an approximation of the original symbols. For measuring the goodness of such an approximation, distortion measures are defined that express the difference between a set of reconstructed samples and the corresponding original samples as a nonnegative real value. A smaller distortion corresponds to a higher approximation quality. A distortion of zero specifies that the reproduced samples are identical to the corresponding original samples.

In this text, we restrict our considerations to the important class of *additive distortion measures*. The distortion between a single reconstructed symbol s' and the corresponding original symbol s is defined as a function $d_1(s, s')$, which satisfies

$$d_1(s, s') \geq 0, \tag{4.1}$$

with equality if and only if $s = s'$. Given such a distortion measure $d_1(s, s')$, the distortion between a set of N reconstructed samples $\boldsymbol{s'} = \{s'_0, s'_1, \ldots, s'_{N-1}\}$ and the corresponding original samples $\boldsymbol{s} = \{s_0, s_1, \ldots, s_{N-1}\}$ is defined by

$$d_N(\boldsymbol{s}, \boldsymbol{s'}) = \frac{1}{N} \sum_{i=0}^{N-1} d_1(s_i, s'_i). \tag{4.2}$$

The most commonly used additive distortion measure is the squared error, $d_1(s, s') = (s - s')^2$. The resulting distortion measure for sets of samples is the *mean squared error* (MSE),

$$d_N(\boldsymbol{s}, \boldsymbol{s'}) = \frac{1}{N} \sum_{i=0}^{N-1} (s_i - s'_i)^2. \tag{4.3}$$

The reasons for the popularity of squared error distortion measures are their simplicity and the mathematical tractability of the associated optimization problems. Throughout this text, we will explicitly use the squared error and mean squared error as distortion measures for single samples and sets of samples, respectively. It should, however, be noted that in most video coding applications the quality of the reconstruction signal is finally judged by human observers. But the MSE does not well correlate with the quality that is perceived by human observers. Nonetheless, MSE-based quality measures are widely used in the video coding community. The investigation of alternative distortion measures for video coding applications is still an active field of research.

In order to evaluate the approximation quality of a code Q, rather than measuring distortion for a given finite symbol sequence, we are interested in a measure for the expected distortion for very long symbol sequences. Given a random process $\boldsymbol{S} = \{S_n\}$, the distortion $\delta(Q)$ associated with a code Q is defined as the limit of the expected distortion as the number of coded symbols approaches infinity,

$$\delta(Q) = \lim_{N \to \infty} E\Big\{ d_N\big(\boldsymbol{S}^{(N)}, \beta_N(\alpha_N(\boldsymbol{S}^{(N)}))\big)\Big\}, \tag{4.4}$$

if the limit exists. $\boldsymbol{S}^{(N)} = \{S_0, S_1, \ldots, S_{N-1}\}$ represents the sequence of the first N random variables of the random process $\boldsymbol{S}$ and $\beta_N(\alpha_N(\cdot))$ specifies the mapping of the first N input symbols to the corresponding reconstructed symbols as given by the code Q.

For stationary processes $\boldsymbol{S}$ with a multivariate pdf $f(\boldsymbol{s})$ and a block code $Q_N = (\alpha_N, \beta_N, \gamma_N)$, the distortion $\delta(Q_N)$ is given by

$$\delta(Q_N) = \int_{\mathcal{R}^N} f(\boldsymbol{s})\, d_N\big(\boldsymbol{s}, \beta_N(\alpha_N(\boldsymbol{s}))\big)\, \mathrm{d}\boldsymbol{s}. \tag{4.5}$$

4.1.2 Rate

Beside the distortion $\delta(Q)$, another important property required for evaluating the performance of a code Q is its *rate*. For coding of a finite symbol sequence $\boldsymbol{s}^{(N)}$, we define the *transmission rate* as the average number of bits per input symbol,

$$r_N(\boldsymbol{s}^{(N)}) = \frac{1}{N}\left|\gamma_N(\alpha_N(\boldsymbol{s}^{(N)}))\right|, \tag{4.6}$$

where $\gamma_N(\alpha_N(\cdot))$ specifies the mapping of the N input symbols to the bit sequence $\boldsymbol{b}^{(\ell)}$ of ℓ bits as given by the code Q and the operator $|\cdot|$ is defined to return the number of bits in the bit sequence that is specified as argument. Similarly as for the distortion, we are interested in a measure for the expected number of bits per symbol for long sequences. For a given random process $\boldsymbol{S} = \{S_n\}$, the *rate* $r(Q)$ associated with a code Q is defined as the limit of the expected number of bits per symbol as the number of transmitted symbols approaches infinity,

$$r(Q) = \lim_{N\to\infty} \frac{1}{N} E\Big\{\left|\gamma_N(\alpha_N(\boldsymbol{S}^{(N)}))\right|\Big\}, \tag{4.7}$$

if the limit exists. For stationary random processes $\boldsymbol{S}$ and a block codes $Q_N = (\alpha_N, \beta_N, \gamma_N)$, the rate $r(Q_N)$ is given by

$$r(Q_N) = \frac{1}{N}\int_{\mathcal{R}^N} f(\boldsymbol{s})\left|\gamma_N(\alpha_N(\boldsymbol{s}))\right| \mathrm{d}\boldsymbol{s}, \tag{4.8}$$

where $f(\boldsymbol{s})$ is the N-th order joint pdf of the random process $\boldsymbol{S}$.

4.1.3 Operational Rate Distortion Function

For a given source $\boldsymbol{S}$, each code Q is associated with a rate distortion point (R, D), which is given by $R = r(Q)$ and $D = \delta(Q)$. In the diagram of Fig. 4.2, the rate distortion points for selected codes are illustrated as dots. The rate distortion plane can be partitioned into a region of achievable rate distortion points and a region of non-achievable rate distortion points. A rate distortion point (R, D) is called *achievable* if there is a code Q with $r(Q) \leq R$ and $\delta(Q) \leq D$. The boundary between the regions of achievable and non-achievable rate distortion points specifies the minimum rate R that is required for representing the source $\boldsymbol{S}$

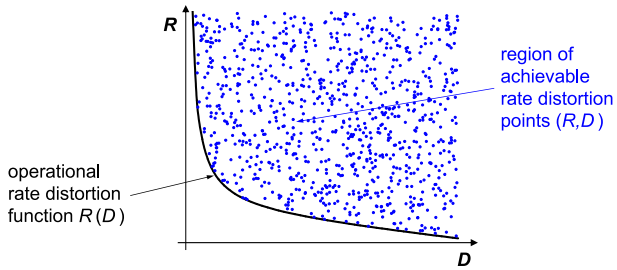


Fig. 4.2 Operational rate distortion function as boundary of the region of achievable rate distortion points. The dots represent rate distortion points for selected codes.

with a distortion less than or equal to a given value D or, alternatively, the minimum distortion D that can be achieved if the source $\boldsymbol{S}$ is coded at a rate less than or equal to a given value R. The function $R(D)$ that describes this fundamental bound for a given source $\boldsymbol{S}$ is called the *operational rate distortion function* and is defined as the infimum of rates $r(Q)$ for all codes Q that achieve a distortion $\delta(Q)$ less than or equal to D,

$$R(D) = \inf_{Q:\,\delta(Q)\leq D} r(Q). \tag{4.9}$$

Fig. 4.2 illustrates the relationship between the region of achievable rate distortion points and the operational rate distortion function. The inverse of the operational rate distortion function is referred to as *operational distortion rate function* $D(R)$ and is defined by

$$D(R) = \inf_{Q:\,r(Q)\leq R} \delta(Q). \tag{4.10}$$

The terms operational rate distortion function and operational distortion rate function are not only used for specifying the best possible performance over all codes Q without any constraints, but also for specifying the performance bound for sets of source codes that are characterized by particular structural or complexity constraints. As an example, such a set of source codes could be the class of scalar quantizers or the class of scalar quantizers with fixed-length codewords. With $\mathcal{G}$ denoting the set of source codes Q with a particular constraint, the operational rate distortion function for a given source $\boldsymbol{S}$ and codes with

the particular constraint is defined by

$$R_{\mathcal{G}}(D) = \inf_{Q \in \mathcal{G}:\, \delta(Q) \leq D} r(Q). \tag{4.11}$$

Similarly, the operational distortion rate function for a given source $\boldsymbol{S}$ and a set $\mathcal{G}$ of codes with a particular constraint is defined by

$$D_{\mathcal{G}}(R) = \inf_{Q \in \mathcal{G}:\, r(Q) \leq R} \delta(Q). \tag{4.12}$$

It should be noted that in contrast to information rate distortion functions, which will be introduced in the next section, operational rate distortion functions are not convex. They are more likely to be step functions, i.e., piecewise constant functions.

4.2 The Information Rate Distortion Function

In the previous section, we have shown that the operational rate distortion function specifies a fundamental performance bound for lossy source coding techniques. But unless we suitably restrict the set of considered codes, it is virtually impossible to determine the operational rate distortion function according to the definition in (4.9). A more accessible expression for a performance bound of lossy codes is given by the information rate distortion function, which was originally introduced by Shannon in [69, 70].

In the following, we first introduce the concept of mutual information before we define the information rate distortion function and investigate its relationship to the operational rate distortion function.

4.2.1 Mutual Information

Although this chapter deals with the lossy coding of random sources, we will introduce the quantity of mutual information for general random variables and vectors of random variables.

Let X and Y be two discrete random variables with alphabets $\mathcal{A}_X = \{x_0, x_1, \cdots, x_{M_X-1}\}$ and $\mathcal{A}_Y = \{y_0, y_1, \cdots, y_{M_Y-1}\}$, respectively. As shown in sec. 3.2, the entropy $H(X)$ represents a lower bound for the average codeword length of a lossless source code for the random variable X. It can also be considered as a measure for the uncertainty

that is associated with the random variable X or as a measure for the average amount of information that is required to describe the random variable X. The conditional entropy $H(X|Y)$ can be interpreted as a measure for the uncertainty that we have about the random variable X if we observe the random variable Y or as the average amount of information that is required to describe the random variable X if the random variable Y is known. The *mutual information* between the discrete random variables X and Y is defined as the difference

$$I(X;Y) = H(X) - H(X|Y). \tag{4.13}$$

The mutual information $I(X;Y)$ is a measure for the reduction of the uncertainty about the random variable X due to the observation of Y. It represents the average amount of information that the random variable Y contains about the random variable X. Inserting the formulas for the entropy (3.13) and conditional entropy (3.20) yields

$$I(X;Y) = \sum_{i=0}^{M_X} \sum_{j=0}^{M_Y} p_{XY}(x_i, y_j) \log_2 \frac{p_{XY}(x_i, y_i)}{p_X(x_i)\, p_Y(y_j)}, \tag{4.14}$$

where p_X and p_Y represent the marginal pmf's of the random variables X and Y, respectively, and p_{XY} denotes the joint pmf.

For extending the concept of mutual information to general random variables we consider two random variables X and Y with marginal the pdf's f_X and f_Y, respectively, and the joint pdf f_{XY}. Either or both of the random variables may be discrete or continuous or of mixed type. Since the entropy, as introduced in sec. 3.2, is only defined for discrete random variables, we investigate the mutual information for discrete approximations X_Δ and Y_Δ of the random variables X and Y.

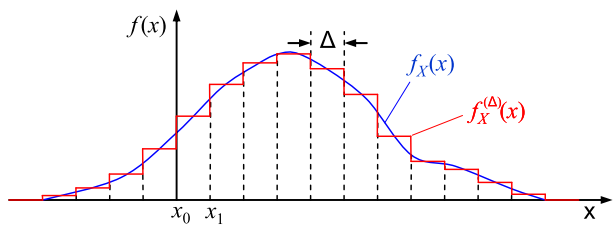


Fig. 4.3 Discretization of a pdf using a quantization step size Δ.

With Δ being a step size, the alphabet of the discrete approximation X_Δ of a random variable X is defined by $\mathcal{A}_{X_\Delta} = \{\ldots, x_{-1}, x_0, x_1, \ldots\}$ with $x_i = i \cdot \Delta$. The event $\{X_\Delta = x_i\}$ is defined to be equal to the event $\{x_i \leq X < x_{i+1}\}$. Furthermore, we define an approximation $f_X^{(\Delta)}$ of the pdf f_X for the random variable X, which is constant inside each half-open interval $[x_i, x_{i+1})$, as illustrated in Fig. 4.3, and is given by

$$\forall x : x_i \leq x < x_{i+1} \qquad f_X^{(\Delta)}(x) = \frac{1}{\Delta} \int_{x_i}^{x_{i+1}} f_X(x')\,\mathrm{d}x'. \tag{4.15}$$

The pmf p_{X_Δ} for the random variable X_Δ can then be expressed as

$$p_{X_\Delta}(x_i) = \int_{x_i}^{x_{i+1}} f_X(x')\,\mathrm{d}x' = f_X^{(\Delta)}(x_i) \cdot \Delta. \tag{4.16}$$

Similarly, we define a piecewise constant approximation $f_{XY}^{(\Delta)}$ for the joint pdf f_{XY} of two random variables X and Y, which is constant inside each 2-dimensional interval $[x_i, x_{i+1}) \times [y_j, y_{j+1})$. The joint pmf $p_{X_\Delta Y_\Delta}$ of the two discrete approximations X_Δ and Y_Δ is then given by

$$p_{X_\Delta Y_\Delta}(x_i, y_j) = f_{XY}^{(\Delta)}(x_i, y_j) \cdot \Delta^2. \tag{4.17}$$

Using the relationships (4.16) and (4.17), we obtain for the mutual information of the discrete random variables X_Δ and Y_Δ

$$I(X_\Delta; Y_\Delta) = \sum_{i=-\infty}^{\infty} \sum_{j=-\infty}^{\infty} f_{XY}^{(\Delta)}(x_i, y_j) \cdot \log_2 \frac{f_{XY}^{(\Delta)}(x_i, y_j)}{f_X^{(\Delta)}(x_i)\, f_Y^{(\Delta)}(y_j)} \cdot \Delta^2. \tag{4.18}$$

If the step size Δ approaches zero, the discrete approximations X_Δ and Y_Δ approach the random variables X and Y. The mutual information $I(X;Y)$ for random variables X and Y can be defined as limit of the mutual information $I(X_\Delta; Y_\Delta)$ as Δ approaches zero,

$$I(X;Y) = \lim_{\Delta \to 0} I(X_\Delta; Y_\Delta). \tag{4.19}$$

If the step size Δ approaches zero, the piecewise constant pdf approximations $f_{XY}^{(\Delta)}$, $f_X^{(\Delta)}$, and $f_Y^{(\Delta)}$ approach the pdf's f_{XY}, f_X, and f_Y, respectively, and the sum in (4.18) approaches the integral

$$I(X;Y) = \int_{-\infty}^{\infty} \int_{-\infty}^{\infty} f_{XY}(x,y)\, \log_2 \frac{f_{XY}(x,y)}{f_X(x)\, f_Y(y)}\,\mathrm{d}x\,\mathrm{d}y, \tag{4.20}$$

which represents the definition of mutual information.

The formula (4.20) shows that the mutual information $I(X;Y)$ is symmetric with respect to the random variables X and Y. The average amount of information that a random variable X contains about another random variable Y is equal to the average amount of information that Y contains about X. Furthermore, the mutual information $I(X;Y)$ is greater than or equal to zero, with equality if and only if $f_{XY}(x,y) = f_X(x)\, f_Y(x), \;\; \forall x, y \in \mathcal{R}$, i.e., if and only if the random variables X and Y are independent. This is a direct consequence of the divergence inequality for probability density functions f and g,

$$-\int_{-\infty}^{\infty} f(s) \log_2 \frac{g(s)}{f(s)} \geq 0, \tag{4.21}$$

which is fulfilled with equality if and only if the pdfs f and g are the same. The divergence inequality can be proved using the inequality $\ln x \geq x - 1$ (with equality if and only if $x = 1$),

$$\begin{aligned} -\int_{-\infty}^{\infty} f(s) \log_2 \frac{g(s)}{f(s)}\, \mathrm{d}s &\geq -\frac{1}{\ln 2} \int_{-\infty}^{\infty} f(s) \left(\frac{g(s)}{f(s)} - 1 \right) \mathrm{d}s \\ &= \frac{1}{\ln 2} \left(\int_{-\infty}^{\infty} f(s)\, \mathrm{d}s - \int_{-\infty}^{\infty} g(s)\, \mathrm{d}s \right) \\ &= 0. \end{aligned} \tag{4.22}$$

For N-dimensional random vectors $\boldsymbol{X} = (X_0, X_1, \ldots, X_{N-1})^T$ and $\boldsymbol{Y} = (Y_0, Y_1, \ldots, Y_{N-1})^T$, the definition of mutual information can be extended according to

$$I(\boldsymbol{X};\boldsymbol{Y}) = \int_{\mathcal{R}^N} \int_{\mathcal{R}^N} f_{\boldsymbol{XY}}(\boldsymbol{x},\boldsymbol{y})\, \log_2 \frac{f_{\boldsymbol{XY}}(\boldsymbol{x},\boldsymbol{y})}{f_{\boldsymbol{X}}(\boldsymbol{x})\, f_{\boldsymbol{Y}}(\boldsymbol{y})}\, \mathrm{d}\boldsymbol{x}\, \mathrm{d}\boldsymbol{y}, \tag{4.23}$$

where $f_{\boldsymbol{X}}$ and $f_{\boldsymbol{Y}}$ denote the marginal pdfs for the random vectors $\boldsymbol{X}$ and $\boldsymbol{Y}$, respectively, and $f_{\boldsymbol{XY}}$ represents the joint pdf.

We now assume that the random vector $\boldsymbol{Y}$ is a discrete random vector and is associated with an alphabet $\mathcal{A}_Y^N$. Then, the pdf $f_{\boldsymbol{Y}}$ and the conditional pdf $f_{\boldsymbol{Y}|\boldsymbol{X}}$ can be written as

$$f_{\boldsymbol{Y}}(\boldsymbol{y}) = \sum_{\boldsymbol{a} \in \mathcal{A}_Y^N} \delta(\boldsymbol{y} - \boldsymbol{a})\, p_{\boldsymbol{Y}}(\boldsymbol{a}), \tag{4.24}$$

$$f_{\boldsymbol{Y}|\boldsymbol{X}}(\boldsymbol{y}|\boldsymbol{x}) = \sum_{\boldsymbol{a} \in \mathcal{A}_Y^N} \delta(\boldsymbol{y} - \boldsymbol{a})\, p_{\boldsymbol{Y}|\boldsymbol{X}}(\boldsymbol{a}|\boldsymbol{x}), \tag{4.25}$$

where $p_{\boldsymbol{Y}}$ denotes the pmf of the discrete random vector $\boldsymbol{Y}$, and $p_{\boldsymbol{Y}|\boldsymbol{X}}$ denotes the conditional pmf of $\boldsymbol{Y}$ given the random vector $\boldsymbol{X}$. Inserting $f_{\boldsymbol{XY}} = f_{\boldsymbol{Y}|\boldsymbol{X}} \cdot f_{\boldsymbol{X}}$ and the expressions (4.24) and (4.25) into the definition (4.23) of mutual information for vectors yields

$$I(\boldsymbol{X};\boldsymbol{Y}) = \int_{\mathcal{R}^N} f_{\boldsymbol{X}}(\boldsymbol{x}) \sum_{\boldsymbol{a}\in\mathcal{A}_Y^N} p_{\boldsymbol{Y}|\boldsymbol{X}}(\boldsymbol{a}|\boldsymbol{x}) \log_2 \frac{p_{\boldsymbol{Y}|\boldsymbol{X}}(\boldsymbol{a}|\boldsymbol{x})}{p_{\boldsymbol{Y}}(\boldsymbol{a})} \, \mathrm{d}\boldsymbol{x}. \tag{4.26}$$

This expression can be re-written as

$$I(\boldsymbol{X};\boldsymbol{Y}) = H(\boldsymbol{Y}) - \int_{-\infty}^{\infty} f_{\boldsymbol{X}}(\boldsymbol{x}) \, H(\boldsymbol{Y}|\boldsymbol{X}=\boldsymbol{x}) \, \mathrm{d}\boldsymbol{x}, \tag{4.27}$$

where $H(\boldsymbol{Y})$ is the entropy of the discrete random vector $\boldsymbol{Y}$ and

$$H(\boldsymbol{Y}|\boldsymbol{X}=\boldsymbol{x}) = -\sum_{\boldsymbol{a}\in\mathcal{A}_Y^N} p_{\boldsymbol{Y}|\boldsymbol{X}}(\boldsymbol{a}|\boldsymbol{x}) \log_2 p_{\boldsymbol{Y}|\boldsymbol{X}}(\boldsymbol{a}|\boldsymbol{x}) \tag{4.28}$$

is the conditional entropy of $\boldsymbol{Y}$ given the event $\{\boldsymbol{X}=\boldsymbol{x}\}$. Since the conditional entropy $H(\boldsymbol{Y}|\boldsymbol{X}=\boldsymbol{x})$ is always nonnegative, we have

$$I(\boldsymbol{X};\boldsymbol{Y}) \leq H(\boldsymbol{Y}). \tag{4.29}$$

Equality is obtained if and only if $H(\boldsymbol{Y}|\boldsymbol{X}=\boldsymbol{x})$ is zero for all $\boldsymbol{x}$ and, hence, if and only if the random vector $\boldsymbol{Y}$ is given by a deterministic function of the random vector $\boldsymbol{X}$.

If we consider two random processes $\boldsymbol{X} = \{X_n\}$ and $\boldsymbol{Y} = \{Y_n\}$ and represent the random variables for N consecutive time instants as random vectors $\boldsymbol{X}^{(N)}$ and $\boldsymbol{Y}^{(N)}$, the mutual information $I(\boldsymbol{X}^{(N)};\boldsymbol{Y}^{(N)})$ between the random vectors $\boldsymbol{X}^{(N)}$ and $\boldsymbol{Y}^{(N)}$ is also referred to as N-th order mutual information and denoted by $I_N(\boldsymbol{X};\boldsymbol{Y})$.

4.2.2 Information Rate Distortion Function

Suppose we have a source $\boldsymbol{S} = \{S_n\}$ that is coded using a lossy source coding system given by a code $Q = (\alpha, \beta, \gamma)$. The output of the lossy coding system can be described by the random process $\boldsymbol{S'} = \{S'_n\}$. Since coding is a deterministic process given by the mapping $\beta(\alpha(\cdot))$, the random process $\boldsymbol{S'}$ describing the reconstructed samples is a deterministic function of the input process $\boldsymbol{S}$. Nonetheless, the statistical

properties of the deterministic mapping given by a code Q can be described by a conditional pdf $g^Q(s'|s) = g_{S'_n|S_n}(s'|s)$. If we consider, as an example, simple scalar quantization, the conditional pdf $g^Q(s'|s)$ represents, for each value of s, a shifted Dirac delta function. In general, $g^Q(s'|s)$ consists of a sum of scaled and shifted Dirac delta functions. Note that the random variables S'_n are always discrete and, hence, the conditional pdf $g^Q(s'|s)$ can also be represented by a conditional pmf. Instead of single samples, we can also consider the mapping of blocks of N successive input samples $\boldsymbol{S}$ to blocks of N successive output samples $\boldsymbol{S'}$. For each value of $N > 0$, the statistical properties of a code Q can then be described by the conditional pdf $g_N^Q(\boldsymbol{s'}|\boldsymbol{s}) = g_{\boldsymbol{S'}|\boldsymbol{S}}(\boldsymbol{s'}|\boldsymbol{s})$.

For the following considerations, we define the N-th order distortion

$$\delta_N(g_N) = \int_{\mathcal{R}^N} \int_{\mathcal{R}^N} f_{\boldsymbol{S}}(\boldsymbol{s})\, g_N(\boldsymbol{s'}|\boldsymbol{s})\, d_N(\boldsymbol{s}, \boldsymbol{s'})\, \mathrm{d}\boldsymbol{s}\, \mathrm{d}\boldsymbol{s'}. \tag{4.30}$$

Given a source $\boldsymbol{S}$, with an N-th order pdf $f_{\boldsymbol{S}}$, and an additive distortion measure d_N, the N-th order distortion $\delta_N(g_N)$ is completely determined by the conditional pdf $g_N = g_{\boldsymbol{S'}|\boldsymbol{S}}$. The distortion $\delta(Q)$ that is associated with a code Q and was defined in (4.4) can be written as

$$\delta(Q) = \lim_{N\to\infty} \delta_N(\, g_N^Q\,). \tag{4.31}$$

Similarly, the N-th order mutual information $I_N(\boldsymbol{S}; \boldsymbol{S'})$ between blocks of N successive input samples and the corresponding blocks of output samples can be written as

$$I_N(g_N) = \int_{\mathcal{R}^N} \int_{\mathcal{R}^N} f_{\boldsymbol{S}}(\boldsymbol{s})\, g_N(\boldsymbol{s'}|\boldsymbol{s})\, \log_2 \frac{g_N(\boldsymbol{s'}|\boldsymbol{s})}{f_{\boldsymbol{S'}}(\boldsymbol{s'})}\, \mathrm{d}\boldsymbol{s}\, \mathrm{d}\boldsymbol{s'}, \tag{4.32}$$

with

$$f_{\boldsymbol{S'}}(\boldsymbol{s'}) = \int_{\mathcal{R}^N} f_{\boldsymbol{S}}(\boldsymbol{s})\, g_N(\boldsymbol{s'}|\boldsymbol{s})\, \mathrm{d}\boldsymbol{s}. \tag{4.33}$$

For a given source $\boldsymbol{S}$, the N-th order mutual information only depends on the N-th order conditional pdf g_N.

We now consider any source code Q with a distortion $\delta(Q)$ that is less than or equal to a given value D. As mentioned above, the output process $\boldsymbol{S'}$ of a source coding system is always discrete. We have shown in sec. 3.3.1 that the average codeword length for lossless coding of a

discrete source cannot be smaller than the entropy rate of the source. Hence, the rate $r(Q)$ of the code Q is greater than or equal to the entropy rate of $\boldsymbol{S'}$,

$$r(Q) \geq \bar{H}(\boldsymbol{S'}). \tag{4.34}$$

By using the definition of the entropy rate $\bar{H}(\boldsymbol{S'})$ in (3.25) and the relationship (4.29), we obtain

$$r(Q) \geq \lim_{N\to\infty} \frac{H_N(\boldsymbol{S})}{N} \geq \lim_{N\to\infty} \frac{I_N(\boldsymbol{S};\boldsymbol{S'})}{N} = \lim_{N\to\infty} \frac{I_N(g_N^Q)}{N}, \tag{4.35}$$

where $H_N(\boldsymbol{S'})$ denotes the block entropy for the random vectors $\boldsymbol{S'}$ of N successive reconstructed samples and $I_N(\boldsymbol{S};\boldsymbol{S'})$ is the mutual information between the N-dimensional random vectors $\boldsymbol{S}$ and the corresponding reconstructions $\boldsymbol{S'}$. A deterministic mapping as given by a source code is a special case of a random mapping. Hence, the N-th order mutual information $I_N(g_N^Q)$ for a particular code Q with $\delta_N(g_N^Q) \leq D$ cannot be smaller than the smallest N-th order mutual information $I_N(g_N)$ that can be achieved using any random mapping $g_N = g_{\boldsymbol{S'}|\boldsymbol{S}}$ with $\delta_N(g_N) \leq D$,

$$I_N(g_N^Q) \geq \inf_{g_N:\, \delta_N(g_N)\leq D} I_N(g_N). \tag{4.36}$$

Consequently, the rate $r(Q)$ is always greater than or equal to

$$R^{(I)}(D) = \lim_{N\to\infty} \inf_{g_N:\, \delta_N(g_N)\leq D} \frac{I_N(g_N)}{N}. \tag{4.37}$$

This fundamental lower bound for all lossy source coding techniques is called the *information rate distortion function*. Every code Q that yields a distortion $\delta(Q)$ less than or equal to any given value D for a source $\boldsymbol{S}$ is associated with a rate $r(Q)$ that is greater than or equal to the information rate distortion function $R^{(I)}(D)$ for the source $\boldsymbol{S}$,

$$\forall Q:\ \delta(Q) \leq D \qquad r(Q) \geq R^{(I)}(D). \tag{4.38}$$

This relationship is called the *fundamental source coding theorem*. The information rate distortion function was first derived by Shannon for iid sources [69, 70] and is for that reason also referred to as *Shannon rate distortion function*.

If we restrict our considerations to iid sources, the N-th order joint pdf $f_{\boldsymbol{S}}(\boldsymbol{s})$ can be represented as the product $\prod_{i=0}^{N-1} f_S(s_i)$ of the marginal pdf $f_S(s)$, with $\boldsymbol{s} = \{s_0, \cdots, s_{N-1}\}$. Hence, for every N, the N-th order distortion $\delta_N(g_N^Q)$ and mutual information $I_N(g_N^Q)$ for a code Q can be expressed using a scalar conditional pdf $g^Q = g_{S'|S}$,

$$\delta_N(g_N^Q) = \delta_1(g^Q) \qquad \text{and} \qquad I_N(g_N^Q) = N \cdot I_1(g^Q). \tag{4.39}$$

Consequently, the information rate distortion function $R^{(I)}(D)$ for iid sources is equal to the so-called *first order information rate distortion function*,

$$R_1^{(I)}(D) = \inf_{g:\, \delta_1(g) \le D} I_1(g). \tag{4.40}$$

In general, the function

$$R_N^{(I)}(D) = \inf_{g_N:\, \delta_N(g_N) \le D} \frac{I_N(g_N)}{N}. \tag{4.41}$$

is referred to as the *N-th order information rate distortion function*. If N approaches infinity, the N-th order information rate distortion function approaches the information rate distortion function,

$$R^{(I)}(D) = \lim_{N \to \infty} R_N^{(I)}(D). \tag{4.42}$$

We have shown that the information rate distortion function represents a fundamental lower bound for all lossy coding algorithms. Using the concept of typical sequences, it can additionally be shown that the information rate distortion function is also asymptotically achievable [4, 22, 11], meaning that for any $\varepsilon > 0$ there exists a code Q with $\delta(Q) \le D$ and $r(Q) \le R^{(I)}(D) + \varepsilon$. Hence, subject to suitable technical assumptions the information rate distortion function is equal to the operational rate distortion function. In the following text, we use the notation $R(D)$ and the term *rate distortion function* to denote both the operational and information rate distortion function. The term operational rate distortion function will mainly be used for denoting the operational rate distortion function for restricted classes of codes.

The inverse of the information rate distortion function is called the *information distortion rate function* or simply the *distortion rate function* and is given by

$$D(R) = \lim_{N \to \infty} \inf_{g_N:\, I_N(g_N)/N \le R} \delta_N(g_N). \tag{4.43}$$

Using this definition, the fundamental source coding theorem (4.38) can also be written as

$$\forall Q : r(Q) \leq R \qquad \delta(Q) \geq D(R). \tag{4.44}$$

The information rate distortion function is defined as a mathematical function of a source. However, an analytical derivation of the information rate distortion function is still very difficult or even impossible, except for some special random processes. An iterative technique for numerically computing close approximations of the rate distortion function for iid sources was developed by Blahut and Arimoto in [6, 3] and is referred to as Blahut-Arimoto algorithm. An overview of the algorithm can be found in [22, 11].

4.2.3 Properties of the Rate Distortion Function

In the following, we state some important properties of the rate distortion function $R(D)$ for the MSE distortion measure[1]. For proofs of these properties, the reader is referred to [4, 22, 11].

- The rate distortion function $R(D)$ is a non-increasing and convex function of D.
- There exists a value $D_{\max}$, so that

  $$\forall D \geq D_{\max} \qquad R(D) = 0. \tag{4.45}$$

  For the MSE distortion measure, the value of $D_{\max}$ is equal to the variance σ^2 of the source.
- For continuous sources $\boldsymbol{S}$, the rate distortion function $R(D)$ approaches infinity as D approaches zero.
- For discrete sources $\boldsymbol{S}$, the minimum rate that is required for a lossless transmission is equal to the entropy rate,

  $$R(0) = \bar{H}(\boldsymbol{S}). \tag{4.46}$$

The last property shows that the fundamental bound for lossless coding is a special case of the fundamental bound for lossy coding.

[1] The properties hold more generally. In particular, all stated properties are valid for additive distortion measures for which the single-letter distortion $d_1(s, s')$ is equal to zero if $s = s'$ and is greater than zero if $s \neq s'$.

4.3 The Shannon Lower Bound

For most random processes, an analytical expression for the rate distortion function cannot be given. In the following, we show how a useful lower bound for the rate distortion function of continuous random processes can be calculated. Before we derive this so-called Shannon lower bound, we introduce the quantity of differential entropy.

4.3.1 Differential Entropy

The mutual information $I(\boldsymbol{X};\boldsymbol{Y})$ of two continuous N-dimensional random vectors $\boldsymbol{X}$ and $\boldsymbol{Y}$ is defined in (4.23). Using the relationship $f_{\boldsymbol{XY}} = f_{\boldsymbol{X}|\boldsymbol{Y}} \cdot f_{\boldsymbol{Y}}$, the integral in this definition can be decomposed into a part that only depends on one of the random vectors and a part that depends on both random vectors,

$$I(\boldsymbol{X};\boldsymbol{Y}) = h(\boldsymbol{X}) - h(\boldsymbol{X}|\boldsymbol{Y}), \tag{4.47}$$

with

$$\begin{aligned} h(\boldsymbol{X}) &= E\{-\log_2 f_{\boldsymbol{X}}(\boldsymbol{X})\} \\ &= -\int_{\mathcal{R}^N} f_{\boldsymbol{X}}(\boldsymbol{x}) \log_2 f_{\boldsymbol{X}}(\boldsymbol{x}) \,\mathrm{d}\boldsymbol{x} \end{aligned} \tag{4.48}$$

and

$$\begin{aligned} h(\boldsymbol{X}|\boldsymbol{Y}) &= E\big\{-\log_2 f_{\boldsymbol{X}|\boldsymbol{Y}}(\boldsymbol{X}|\boldsymbol{Y})\big\} \\ &= -\int_{\mathcal{R}^N}\int_{\mathcal{R}^N} f_{\boldsymbol{XY}}(\boldsymbol{x},\boldsymbol{y}) \log_2 f_{\boldsymbol{X}|\boldsymbol{Y}}(\boldsymbol{x}|\boldsymbol{y}) \,\mathrm{d}\boldsymbol{x}\,\mathrm{d}\boldsymbol{y}. \end{aligned} \tag{4.49}$$

In analogy to the discrete entropy introduced in Chapter 3, the quantity $h(\boldsymbol{X})$ is called the *differential entropy* of the random vector $\boldsymbol{X}$ and the quantity $h(\boldsymbol{X}|\boldsymbol{Y})$ is referred to as *conditional differential entropy* of the random vector $\boldsymbol{X}$ given the random vector $\boldsymbol{Y}$.

Since $I(\boldsymbol{X};\boldsymbol{Y})$ is always nonnegative, we can conclude that conditioning reduces the differential entropy,

$$h(\boldsymbol{X}|\boldsymbol{Y}) \leq h(\boldsymbol{X}), \tag{4.50}$$

similarly as conditioning reduces the discrete entropy.

For continuous random processes $\boldsymbol{S} = \{S_n\}$, the random variables S_n for N consecutive time instants can be represented as a random vector $\boldsymbol{S}^{(N)} = (S_0, \cdots, S_{N-1})^T$. The differential entropy $h(\boldsymbol{S}^{(N)})$ for the vectors $\boldsymbol{S}^{(N)}$ is then also referred to as *N-th order differential entropy* and is denoted by

$$h_N(\boldsymbol{S}) = h(\boldsymbol{S}^{(N)}) = h(S_0, \cdots, S_{N-1}) \tag{4.51}$$

If, for a continuous random process $\boldsymbol{S}$, the limit

$$\bar{h}(\boldsymbol{S}) = \lim_{N\to\infty} \frac{h_N(\boldsymbol{S})}{N} = \lim_{N\to\infty} \frac{h(S_0, \cdots, S_{N-1})}{N} \tag{4.52}$$

exists, it is called the *differential entropy rate* of the process $\boldsymbol{S}$.

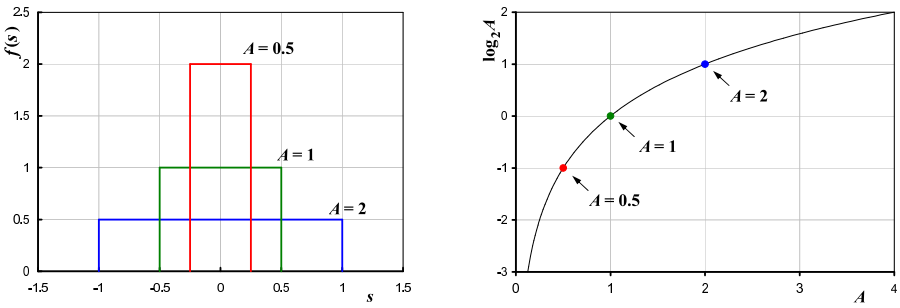


Fig. 4.4 Probability density function and differential entropy for uniform distributions.

The differential entropy has a different meaning than the discrete entropy. This can be illustrated by considering an iid process $\boldsymbol{S} = \{S_n\}$ with a uniform pdf $f(s)$, with $f(s) = 1/A$ for $|s| \leq A/2$ and $f(s) = 0$ for $|s| > A/2$. The first order differential entropy for this process is

$$h(S) = -\int_{-A/2}^{A/2} \frac{1}{A} \log_2 \frac{1}{A} \, \mathrm{d}s = \log_2 A \, \frac{1}{A} \int_{-A/2}^{A/2} \mathrm{d}s = \log_2 A. \tag{4.53}$$

In Fig. 4.4, the differential entropy $h(S)$ for the uniform iid process is shown as function of the parameter A. In contrast to the discrete entropy, the differential entropy can be either positive or negative. The discrete entropy is only finite for discrete alphabet sources, it is infinite for continuous alphabet sources. The differential entropy, however, is mainly useful for continuous random processes. For discrete random processes, it can be considered to be minus infinity.

As an example, we consider a stationary Gaussian random process with a mean μ and an N-th order autocovariance matrix $\boldsymbol{C}_N$. The N-th order pdf $f_G(\boldsymbol{s})$ is given in (2.51), where $\boldsymbol{\mu}_N$ represents a vector with all N elements being equal to the mean μ. For the N-th order differential entropy $h_N^{(G)}$ of the stationary Gaussian process, we obtain

$$\begin{aligned} h_N^{(G)}(\boldsymbol{S}) &= -\int_{\mathcal{R}^N} f_G(\boldsymbol{s})\, \log_2 f_G(\boldsymbol{s})\, \mathrm{d}\boldsymbol{s} \\ &= \frac{1}{2}\log_2\big(\,(2\pi)^N|\boldsymbol{C}_N|\,\big) \\ &\quad + \frac{1}{2\ln 2}\int_{\mathcal{R}^N} f_G(\boldsymbol{s})\, (\boldsymbol{s}-\boldsymbol{\mu}_N)^T\boldsymbol{C}_N^{-1}(\boldsymbol{s}-\boldsymbol{\mu}_N)\, \mathrm{d}\boldsymbol{s}. \end{aligned} \tag{4.54}$$

By reformulating the matrix multiplication in the last integral as sum, it can be shown that for any random process with an N-th order pdf $f(\boldsymbol{s})$ and an N-th order autocovariance matrix $\boldsymbol{C}_N$,

$$\int_{\mathcal{R}^N} f(\boldsymbol{s})(\boldsymbol{s}-\boldsymbol{\mu}_N)^T\boldsymbol{C}_N^{-1}(\boldsymbol{s}-\boldsymbol{\mu}_N)\, \mathrm{d}\boldsymbol{s} = N. \tag{4.55}$$

A step-by-step derivation of this result can be found in [11]. Inserting (4.55) into (4.54) and using $\log_2 e = (\ln 2)^{-1}$ yields

$$\begin{aligned} h_N^{(G)}(\boldsymbol{S}) &= \frac{1}{2}\log_2\big(\,(2\pi)^N|\boldsymbol{C}_N|\,\big) + \frac{N}{2}\log_2 e \\ &= \frac{1}{2}\log_2\big(\,(2\pi e)^N|\boldsymbol{C}_N|\,\big). \end{aligned} \tag{4.56}$$

Now, we consider any stationary random process $\boldsymbol{S}$ with a mean μ and an N-th order autocovariance matrix $\boldsymbol{C}_N$. The N-th order pdf of this process is denoted by $f(\boldsymbol{s})$. Using the divergence inequality (4.21), we obtain for its N-th order differential entropy,

$$\begin{aligned} h_N(\boldsymbol{S}) &= -\int_{\mathcal{R}^N} f(\boldsymbol{s})\, \log_2 f(\boldsymbol{s})\, \mathrm{d}\boldsymbol{s} \\ &\le -\int_{\mathcal{R}^N} f(\boldsymbol{s})\, \log_2 f_G(\boldsymbol{s})\, \mathrm{d}\boldsymbol{s} \\ &= \frac{1}{2}\log_2\big(\,(2\pi)^N|\boldsymbol{C}_N|\,\big) \\ &\quad + \frac{1}{2\ln 2}\int_{\mathcal{R}^N} f(\boldsymbol{s})(\boldsymbol{s}-\boldsymbol{\mu}_N)^T\boldsymbol{C}_N^{-1}(\boldsymbol{s}-\boldsymbol{\mu}_N)\, \mathrm{d}\boldsymbol{s}, \end{aligned} \tag{4.57}$$

where $f_G(\boldsymbol{s})$ represents the N-th order pdf of the stationary Gaussian process with the same mean μ and the same N-th order autocovariance matrix $\boldsymbol{C}_N$. Inserting (4.55) and (4.56) yields

$$h_N(\boldsymbol{S}) \leq h_N^{(G)}(\boldsymbol{S}) = \frac{1}{2}\log_2\big((2\pi e)^N|\boldsymbol{C}_N|\big). \tag{4.58}$$

Hence, the N-th order differential entropy of any stationary non-Gaussian process is less than the N-th order differential entropy of a stationary Gaussian process with the same N-th order autocovariance matrix $\boldsymbol{C}_N$.

As shown in (4.56), the N-th order differential entropy of a stationary Gaussian process depends on the determinant of its N-th order autocovariance matrix $|\boldsymbol{C}_N|$. The determinant $|\boldsymbol{C}_N|$ is given by the product of the eigenvalues ξ_i of the matrix $\boldsymbol{C}_N$, $|\boldsymbol{C}_N| = \prod_{i=0}^{N-1}\xi_i$. The trace of the N-th order autocovariance matrix $\mathrm{tr}(\boldsymbol{C}_N)$ is given by the sum of its eigenvalues, $\mathrm{tr}(\boldsymbol{C}_N) = \sum_{i=0}^{N-1}\xi_i$, and, according to (2.39), also by $\mathrm{tr}(\boldsymbol{C}_N) = N\cdot\sigma^2$, with σ^2 being the variance of the Gaussian process. Hence, for a given variance σ^2, the sum of the eigenvalues is constant. With the inequality of arithmetic and geometric means,

$$\left(\prod_{i=0}^{N-1} x_i\right)^{\frac{1}{N}} \leq \frac{1}{N}\sum_{i=0}^{N-1} x_i, \tag{4.59}$$

which holds with equality if and only if $x_0 = x_1 = \ldots = x_{N-1}$, we obtain the inequality

$$|\boldsymbol{C}_N| = \prod_{i=0}^{N-1}\xi_i \leq \left(\frac{1}{N}\sum_{i=0}^{N-1}\xi_i\right)^N = \sigma^{2N}. \tag{4.60}$$

Equality holds if and only if all eigenvalues of $\boldsymbol{C}_N$ are the same, i.e, if and only if the Gaussian process is iid. Consequently, the N-th order differential entropy of a stationary process $\boldsymbol{S}$ with a variance σ^2 is bounded by

$$h_N(\boldsymbol{S}) \leq \frac{N}{2}\log_2\big(2\pi e\sigma^2\big). \tag{4.61}$$

It is maximized if and only if the process is a Gaussian iid process.

4.3.2 Shannon Lower Bound

Using the relationship (4.47) and the notation $I_N(g_N) = I_N(\boldsymbol{S}; \boldsymbol{S'})$, the rate distortion function $R(D)$ defined in (4.37) can be written as

$$\begin{aligned} R(D) &= \lim_{N\to\infty} \inf_{g_N:\, \delta_N(g_N)\le D} \frac{I_N(\boldsymbol{S}; \boldsymbol{S'})}{N} \\ &= \lim_{N\to\infty} \inf_{g_N:\, \delta_N(g_N)\le D} \frac{h_N(\boldsymbol{S}) - h_N(\boldsymbol{S}|\boldsymbol{S'})}{N} \\ &= \lim_{N\to\infty} \frac{h_N(\boldsymbol{S})}{N} - \lim_{N\to\infty} \sup_{g_N:\, \delta_N(g_N)\le D} \frac{h_N(\boldsymbol{S}|\boldsymbol{S'})}{N} \\ &= \bar{h}(\boldsymbol{S}) - \lim_{N\to\infty} \sup_{g_N:\, \delta_N(g_N)\le D} \frac{h_N(\boldsymbol{S} - \boldsymbol{S'}|\boldsymbol{S'})}{N}, \end{aligned} \tag{4.62}$$

where the subscripts N indicate the N-th order mutual information and differential entropy. The last equality follows from the fact that the differential entropy is independent of the mean of a given pdf.

Since conditioning reduces the differential entropy, as has been shown in (4.50), the rate distortion function is bounded by

$$R(D) \ge R_L(D), \tag{4.63}$$

with

$$R_L(D) = \bar{h}(\boldsymbol{S}) - \lim_{N\to\infty} \sup_{g_N:\, \delta_N(g_N)\le D} \frac{h_N(\boldsymbol{S} - \boldsymbol{S'})}{N}. \tag{4.64}$$

The lower bound $R_L(D)$ is called the *Shannon lower bound* (SLB).

For stationary processes and the MSE distortion measure, the distortion $\delta_N(g_N)$ in (4.64) is equal to the variance σ_Z^2 of the process $\boldsymbol{Z} = \boldsymbol{S} - \boldsymbol{S'}$. Furthermore, we have shown in (4.61) that the maximum N-th order differential entropy for a stationary process with a given variance σ_Z^2 is equal to $\frac{N}{2}\log_2(2\pi e\,\sigma_Z^2)$. Hence, the Shannon lower bound for stationary processes and MSE distortion is given by

$$R_L(D) = \bar{h}(\boldsymbol{S}) - \frac{1}{2}\log_2\big(2\pi e D\big). \tag{4.65}$$

Since we concentrate on the MSE distortion measure in this text, we call $R_L(D)$ given in (4.65) the Shannon lower bound in the following without mentioning that it is only valid for the MSE distortion measure.

Shannon Lower Bound for IID Sources. The N-th order differential entropy for iid sources $\boldsymbol{S} = \{S_n\}$ is equal to

$$h_N(\boldsymbol{S}) = E\{-\log_2 f_{\boldsymbol{S}}(\boldsymbol{S})\} = \sum_{i=0}^{N-1} E\{-\log_2 f_S(S_n)\} = N \cdot h(S), \quad (4.66)$$

where $h(S)$ denotes the first order differential entropy. Hence, the Shannon lower bound for iid sources is given by

$$R_L(D) = h(S) - \frac{1}{2}\log_2\big(2\pi e D\big), \quad (4.67)$$

$$D_L(R) = \frac{1}{2\pi e} \cdot 2^{2h(S)} \cdot 2^{-2R}. \quad (4.68)$$

In the following, the differential entropy $h(S)$ and the Shannon lower bound $D_L(R)$ are given for three distributions. For the example of the Laplacian iid process with $\sigma^2 = 1$, Fig. 4.5 compares the Shannon lower bound $D_L(R)$ with the distortion rate function $D(R)$, which was calculated using the Blahut-Arimoto algorithm [6, 3].

Uniform pdf:

$$h(S) = \frac{1}{2}\log_2(12\sigma^2) \quad \Rightarrow \quad D_L(R) = \frac{6}{\pi e} \cdot \sigma^2 \cdot 2^{-2R} \quad (4.69)$$

Laplacian pdf:

$$h(S) = \frac{1}{2}\log_2(2e^2\sigma^2) \quad \Rightarrow \quad D_L(R) = \frac{e}{\pi} \cdot \sigma^2 \cdot 2^{-2R} \quad (4.70)$$

Gaussian pdf:

$$h(S) = \frac{1}{2}\log_2(2\pi e\sigma^2) \quad \Rightarrow \quad D_L(R) = \sigma^2 \cdot 2^{-2R} \quad (4.71)$$

Asymptotic Tightness. The comparison of the Shannon lower bound $D_L(R)$ and the distortion rate function $D(R)$ for the Laplacian iid source in Fig. 4.5 indicates that the Shannon lower bound approaches the distortion rate function for small distortions or high rates. For various distortion measures, including the MSE distortion, it can in fact be shown that the Shannon lower bound approaches the rate distortion function as the distortion approaches zero,

$$\lim_{D \to 0} R(D) - R_L(D) = 0. \quad (4.72)$$

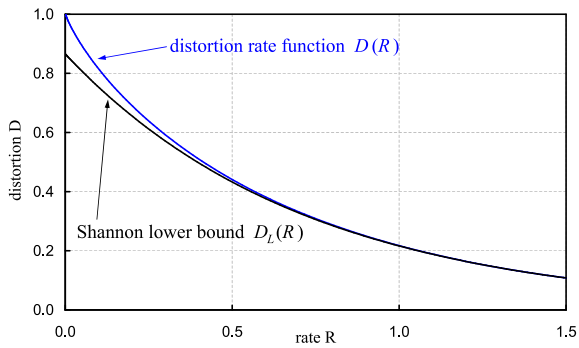


Fig. 4.5 Comparison of the Shannon lower bound $D_L(R)$ and the distortion rate function $D(R)$ for a Laplacian iid source with unit variance ($\sigma^2 = 1$).

Consequently, the Shannon lower bound represents a suitable reference for the evaluation of lossy coding techniques at high rates or small distortions. Proofs for the asymptotic tightness of the Shannon lower bound for various distortion measures can be found in [48, 5, 47].

Shannon Lower Bound for Gaussian Sources. For sources with memory, an exact analytic derivation of the Shannon lower bound is usually not possible. One of the few examples for which the Shannon lower bound can be expressed analytically is the stationary Gaussian process. The N-th order differential entropy for a stationary Gaussian process has been derived in (4.56). Inserting this result into the definition of the Shannon lower bound (4.65) yields

$$R_L(D) = \lim_{N\to\infty} \frac{1}{2N} \log_2 |\boldsymbol{C}_N| - \frac{1}{2} \log_2 D, \tag{4.73}$$

where $\boldsymbol{C}_N$ is the N-th order autocorrelation matrix. The determinant of a matrix is given by the product of its eigenvalues. With $\xi_i^{(N)}$, for $i = 0, 1, \ldots, N-1$, denoting the N eigenvalues of the N-th order autocorrelation matrix $\boldsymbol{C}_N$, we obtain

$$R_L(D) = \lim_{N\to\infty} \frac{1}{2N} \sum_{i=0}^{N-1} \log_2 \xi_i^{(N)} - \frac{1}{2} \log_2 D. \tag{4.74}$$

In order to proceed, we restrict our considerations to Gaussian processes with zero mean, in which case the autocovariance matrix $\boldsymbol{C}_N$ is

equal to the autocorrelation matrix $\boldsymbol{R}_N$, and apply GRENANDER and SZEGÖ's theorem [29] for sequences of Toeplitz matrices. For a review of Toeplitz matrices, including the theorem for sequences of Toeplitz matrices, we recommend the tutorial [23]. GRENANDER and SZEGÖ's theorem can be stated as follows:

> If $\boldsymbol{R}_N$ is a sequence of Hermitian Toeplitz matrices with elements ϕ_k on the k-th diagonal, the infimum $\Phi_{\text{inf}} = \inf_\omega \Phi(\omega)$ and supremum $\Phi_{\text{sup}} = \sup_\omega \Phi(\omega)$ of the Fourier series
>
> $$\Phi(\omega) = \sum_{k=-\infty}^{\infty} \phi_k \, e^{-j\omega k} \tag{4.75}$$
>
> are finite, and the function G is continuous in the interval $[\Phi_{\text{inf}}, \Phi_{\text{sup}}]$, then
>
> $$\lim_{N\to\infty} \frac{1}{N} \sum_{i=0}^{N-1} G\left(\xi_i^{(N)}\right) = \frac{1}{2\pi} \int_{-\pi}^{\pi} G\left(\Phi(\omega)\right) \, \mathrm{d}\omega, \tag{4.76}$$
>
> where $\xi_i^{(N)}$, for $i = 0, 1, \ldots, N-1$, denote the eigenvalues of the N-th matrix $\boldsymbol{R}_N$.

A matrix is called Hermitian if it is equal to its conjugate transpose. This property is always fulfilled for real symmetric matrices as the autocorrelation matrices of stationary processes. Furthermore, the Fourier series (4.75) for the elements of the autocorrelation matrix $\boldsymbol{R}_N$ is the *power spectral density* $\Phi_{SS}(\omega)$. If we assume that the power spectral density is finite and greater than 0 for all frequencies ω, the limit in (4.74) can be replaced by an integral according to (4.76). The Shannon lower bound $R_L(D)$ of a stationary Gaussian process with zero-mean and a power spectral density $\Phi_{SS}(\omega)$ is given by

$$R_L(D) = \frac{1}{4\pi} \int_{-\pi}^{\pi} \log_2 \frac{\Phi_{SS}(\omega)}{D} \, \mathrm{d}\omega. \tag{4.77}$$

A nonzero mean does not have any impact on the Shannon lower bound $R_L(D)$, but on the power spectral density $\Phi_{SS}(\omega)$.

For a stationary zero-mean Gauss-Markov process, the entries of the autocorrelation matrix are given by $\phi_k = \sigma^2 \rho^{|k|}$, where σ^2 is the signal variance and ρ is the correlation coefficient between successive samples. Using the relationship $\sum_{k=1}^{\infty} a^k \, e^{-jkx} = a/(e^{-jx} - a)$, we obtain

$$\begin{aligned}\Phi_{SS}(\omega) &= \sum_{k=-\infty}^{\infty} \sigma^2 \, \rho^{|k|} \, e^{-j\omega k} = \sigma^2 \left(1 + \frac{\rho}{e^{-j\omega} - \rho} + \frac{\rho}{e^{j\omega} - \rho}\right) \\ &= \sigma^2 \frac{1 - \rho^2}{1 - 2\rho \cos\omega + \rho^2}. \end{aligned} \tag{4.78}$$

Inserting this relationship into (4.77) yields

$$\begin{aligned} R_L(D) &= \frac{1}{4\pi} \int_{-\pi}^{\pi} \log_2 \frac{\sigma^2 (1 - \rho^2)}{D} \, \mathrm{d}\omega - \frac{1}{4\pi} \underbrace{\int_{-\pi}^{\pi} \log_2 (1 - 2\rho \cos\omega + \rho^2) \, \mathrm{d}\omega}_{=0} \\ &= \frac{1}{2} \log_2 \frac{\sigma^2 \, (1 - \rho^2)}{D}, \end{aligned} \tag{4.79}$$

where we used $\int_0^{\pi} \ln(a^2 - 2ab \cos x + b^2) \, \mathrm{d}x = 2\pi \ln a$, for $a \geq b > 0$. As discussed above, the mean of a stationary process does not have any impact on the Shannon rate distortion function or the Shannon lower bound. Hence, the distortion rate function $D_L(R)$ for the Shannon lower bound of a stationary Gauss-Markov process with a variance σ^2 and a correlation coefficient ρ is given by

$$D_L(R) = (1 - \rho^2) \, \sigma^2 \, 2^{-2R}. \tag{4.80}$$

This result can also be obtained by directly inserting the formula (2.50) for the determinant $|\boldsymbol{C}_N|$ of the N-th order autocovariance matrix for Gauss-Markov processes into the expression (4.73).

4.4 Rate Distortion Function for Gaussian Sources

Stationary Gaussian sources play a fundamental role in rate distortion theory. We have shown that the Gaussian source maximize the differential entropy, and thus also the Shannon lower bound, for a given variance or autocovariance function. Stationary Gaussian sources are also one of the few examples, for which the rate distortion function can be exactly derived.

4.4.1 Gaussian IID Sources

Before stating another important property of Gaussian iid sources, we calculate their rate distortion function. Therefore, we first derive a lower bound and then show that this lower bound is achievable. To prove that the lower bound is achievable, it is sufficient to show that there is a conditional pdf $g_{S'|S}(s'|s)$ for which the mutual information $I_1(g_{S'|S})$ is equal to the lower bound for a given distortion D.

The Shannon lower bound for Gaussian iid sources as distortion rate function $D_L(R)$ has been derived in sec. 4.3. The corresponding rate distortion function is given by

$$R_L(D) = \frac{1}{2}\log_2 \frac{\sigma^2}{D}, \tag{4.81}$$

where σ^2 is the signal variance. For proving that the rate distortion function is achievable, it is more convenient to look at the pdf of the reconstruction $f_{S'}(s')$ and the conditional pdf $g_{S|S'}(s|s')$ of the input given the reconstruction.

For distortions $D < \sigma^2$, we choose

$$f_{S'}(s') = \frac{1}{\sqrt{2\pi(\sigma^2 - D)}}\, e^{-\frac{(s'-\mu)^2}{2\,(\sigma^2-D)}}, \tag{4.82}$$

$$g_{S|S'}(s|s') = \frac{1}{\sqrt{2\pi D}}\, e^{-\frac{(s-s')^2}{2\,D}}, \tag{4.83}$$

where μ denotes the mean of the Gaussian iid process. It should be noted that the conditional pdf $g_{S|S'}$ represents a Gaussian pdf for the random variables $Z_n = S_n - S'_n$, which are given by the difference of the corresponding random variables S_n and S'_n. We now verify that the pdf $f_S(s)$ that we obtain with the choices (4.82) and (4.83) represents the Gaussian pdf with a mean μ and a variance σ^2. Since the random variables S_n can be represented as the sum $S'_n + Z_n$, the pdf $f_S(s)$ is given by the convolution of $f_{S'}(s')$ and $g_{S|S'}(s|s')$. And since means and variances add when normal densities are convolved, the pdf $f_S(s)$ that is obtained is a Gaussian pdf with a mean $\mu = \mu + 0$ and a variance $\sigma^2 = (\sigma^2 - D) + D$. Hence, the choices (4.82) and (4.83) are valid, and

the conditional pdf $g_{S'|S}(s'|s)$ could be calculated using Bayes rule

$$g_{S'|S}(s'|s) = g_{S|S'}(s|s')\,\frac{f_{S'}(s')}{f_S(s)}. \tag{4.84}$$

The resulting distortion is given by the variance of the difference process $Z_n = S_n - S'_n$,

$$\delta_1(g_{S'|S}) = E\big\{(S_n - S'_n)^2\big\} = E\big\{Z_n^2\big\} = D. \tag{4.85}$$

For the mutual information, we obtain

$$\begin{aligned} I_1(g_{S'|S}) &= h(S_n) - h(S_n|S'_n) = h(S_n) - h(S_n - S'_n) \\ &= \frac{1}{2}\log_2\big(2\pi e\sigma^2\big) - \frac{1}{2}\log_2\big(2\pi e D\big) = \frac{1}{2}\log_2\frac{\sigma^2}{D}. \end{aligned} \tag{4.86}$$

Here, we used the fact that the conditional pdf $g_{S|S'}(s|s')$ only depends on the difference $s - s'$ as given by the choice (4.83).

The results show that, for any distortion $D < \sigma^2$, we can find a conditional pdf $g_{S'|S}$ that achieves the Shannon lower bound. For greater distortions, we choose $g_{S'|S}(s'|s) = \delta(0)$, which gives a distortion of σ^2 and a rate of zero. Consequently, the rate distortion function for Gaussian iid sources is given by

$$R(D) = \begin{cases} \frac{1}{2}\log_2\frac{\sigma^2}{D} & : \quad D < \sigma^2 \\ 0 & : \quad D \geq \sigma^2 \end{cases}. \tag{4.87}$$

The corresponding distortion rate function is given by

$$D(R) = \sigma^2\, 2^{-2R}. \tag{4.88}$$

It is important to note that the rate distortion function for a Gaussian iid process is equal to the Shannon lower bound for the entire range of rates. Furthermore, it can be shown [4] that for every iid process with a given variance σ^2, the rate distortion function lies below that of the Gaussian iid process with the same variance.

4.4.2 Gaussian Sources with Memory

For deriving the rate distortion function $R(D)$ for a stationary Gaussian process with memory, we decompose it into a number N of independent stationary Gaussian sources. The N-th order rate distortion

function $R_N(D)$ can then be expressed using the rate distortion function for Gaussian iid processes and the rate distortion function $R(D)$ is obtained by considering the limit of $R_N(D)$ as N approaches infinity.

As we stated in sec. 2.3, the N-th order pdf of a stationary Gaussian process is given by

$$f_{\boldsymbol{S}}(\boldsymbol{s}) = \frac{1}{(2\pi)^{N/2}\,|\boldsymbol{C}_N|^{1/2}}\; e^{-\frac{1}{2}(\boldsymbol{s}-\boldsymbol{\mu}_N)^T \boldsymbol{C}_N^{-1}(\boldsymbol{s}-\boldsymbol{\mu}_N)} \tag{4.89}$$

where $\boldsymbol{s}$ is a vector of N consecutive samples, $\boldsymbol{\mu}_N$ is a vector with all N elements being equal to the mean μ, and $\boldsymbol{C}_N$ is the N-th order autocovariance matrix. Since $\boldsymbol{C}_N$ is a symmetric and real matrix, it has N real eigenvalues $\xi_i^{(N)}$, for $i = 0, 1, \ldots, N-1$. The eigenvalues are solutions of the equation

$$\boldsymbol{C}_N \cdot \boldsymbol{v}_i^{(N)} = \xi_i^{(N)} \cdot \boldsymbol{v}_i^{(N)}, \tag{4.90}$$

where $\boldsymbol{v}_i^{(N)}$ represents a nonzero vector with unit norm, which is called a unit-norm eigenvector corresponding to the eigenvalue $\xi_i^{(N)}$. Let $\boldsymbol{A}_N$ be the matrix whose columns are build by the N unit-norm eigenvectors,

$$\boldsymbol{A}_N = \left(\boldsymbol{v}_0^{(N)}, \boldsymbol{v}_1^{(N)}, \cdots, \boldsymbol{v}_{N-1}^{(N)} \right). \tag{4.91}$$

By combining the N equations (4.90) for $i = 0, 1, \ldots, N-1$, we obtain the matrix equation

$$\boldsymbol{C}_N\, \boldsymbol{A}_N = \boldsymbol{A}_N\, \Xi_N, \tag{4.92}$$

where

$$\boldsymbol{\Xi}_N = \begin{pmatrix} \xi_0^{(N)} & 0 & \ldots & 0 \\ 0 & \xi_1^{(N)} & \ldots & 0 \\ \vdots & \vdots & \ddots & 0 \\ 0 & 0 & 0 & \xi_{N-1}^{(N)} \end{pmatrix} \tag{4.93}$$

is a diagonal matrix that contains the N eigenvalues of $\boldsymbol{C}_N$ on its main diagonal. The eigenvectors are orthogonal to each other and $\boldsymbol{A}_N$ is an orthogonal matrix.

Given the stationary Gaussian source $\{S_n\}$, we construct a source $\{U_n\}$ by decomposing the source $\{S_n\}$ into vectors $\boldsymbol{S}$ of N successive random variables and applying the transform

$$\boldsymbol{U} = \boldsymbol{A}_N^{-1}\,(\boldsymbol{S} - \boldsymbol{\mu}_N) = \boldsymbol{A}_N^T\,(\boldsymbol{S} - \boldsymbol{\mu}_N) \tag{4.94}$$

to each of these vectors. Since $\boldsymbol{A}_N$ is orthogonal, its inverse $\boldsymbol{A}^{-1}$ exists and is equal to its transpose $\boldsymbol{A}^T$. The resulting source $\{U_n\}$ is given by the concatenation of the random vectors $\boldsymbol{U}$. Similarly, the inverse transform for the reconstructions $\{U'_n\}$ and $\{S'_n\}$ is given by

$$\boldsymbol{S'} = \boldsymbol{A}_N \, \boldsymbol{U'} + \boldsymbol{\mu}_N, \tag{4.95}$$

with $\boldsymbol{U'}$ and $\boldsymbol{S'}$ denoting the corresponding vectors of N successive random variables. Since the coordinate mapping (4.95) is the inverse of the mapping (4.94), the N-th order mutual information $I_N(\boldsymbol{U};\boldsymbol{U'})$ is equal to the N-th order mutual information $I_N(\boldsymbol{S};\boldsymbol{S'})$. A proof of this statement can be found in [4]. Furthermore, since $\boldsymbol{A}_N$ is orthogonal, the transform

$$\big(\boldsymbol{U'} - \boldsymbol{U}\big) = \boldsymbol{A}_N\big(\boldsymbol{S'} - \boldsymbol{S}\big) \tag{4.96}$$

preserves the Euclidean norm[2]. The MSE distortion between any realization $\boldsymbol{s}$ of the random vector $\boldsymbol{S}$ and its reconstruction $\boldsymbol{s'}$

$$d_N(\boldsymbol{s};\boldsymbol{s'}) = \frac{1}{N}\sum_{i=0}^{N-1}(s_i - s'_i)^2 = \frac{1}{N}\sum_{i=0}^{N-1}(u_i - u'_i)^2 = d_N(\boldsymbol{u};\boldsymbol{u'}) \tag{4.97}$$

is equal to the distortion between the corresponding vector $\boldsymbol{u}$ and its reconstruction $\boldsymbol{u'}$. Hence, the N-th order rate distortion function $R_N(D)$ for the stationary Gaussian source $\{S_n\}$ is equal to the N-th order rate distortion function for the random process $\{U_n\}$.

A linear transformation of a Gaussian random vector results in another Gaussian random vector. For the mean vector and the autocorrelation matrix of $\boldsymbol{U}$, we obtain

$$E\{\boldsymbol{U}\} = \boldsymbol{A}_N^T\,(E\{\boldsymbol{S}\} - \boldsymbol{\mu}_N) = \boldsymbol{A}_N^T\,(\boldsymbol{\mu}_N - \boldsymbol{\mu}_N) = \boldsymbol{0} \tag{4.98}$$

and

$$\begin{aligned} E\big\{\boldsymbol{U}\,\boldsymbol{U}^T\big\} &= \boldsymbol{A}_N^T\,E\Big\{(\boldsymbol{S} - \boldsymbol{\mu}_N)\,(\boldsymbol{S} - \boldsymbol{\mu}_N)^T\Big\}\,\boldsymbol{A}_N \\ &= \boldsymbol{A}_N^T\,\boldsymbol{C}_N\,\boldsymbol{A}_N = \boldsymbol{\Xi}_N. \end{aligned} \tag{4.99}$$

[2] We will show in sec. 7.2 that every orthogonal transform preserves the MSE distortion.

Since $\boldsymbol{\Xi}_N$ is a diagonal matrix, the pdf of the random vectors $\boldsymbol{U}$ is given by the product

$$f_{\boldsymbol{U}}(\boldsymbol{u}) = \frac{1}{(2\pi)^{N/2}\,|\boldsymbol{\Xi}_N|^{1/2}}\, e^{-\frac{1}{2}\boldsymbol{u}^T\boldsymbol{\Xi}_N^{-1}\boldsymbol{u}} = \prod_{i=0}^{N-1} \frac{1}{\sqrt{2\pi\xi_i^{(N)}}}\, e^{-\frac{u_i^2}{2\,\xi_i^{(N)}}} \tag{4.100}$$

of the pdf's of the Gaussian components U_i. Consequently, the components U_i are independent of each other.

In sec. 4.2.2, we have shown how the N-th order mutual information and the N-th order distortion for a code Q can be described by a conditional pdf $g_N^Q = g_{\boldsymbol{U}'|\boldsymbol{U}}$ that characterizes the mapping of the random vectors $\boldsymbol{U}$ onto the corresponding reconstruction vectors $\boldsymbol{U}'$. Due to the independence of the components U_i of the random vectors $\boldsymbol{U}$, the N-th order mutual information $I_N(g_N^Q)$ and the N-th order distortion $\delta_N(g_N^Q)$ for a code Q can be written as

$$I_N(g_N^Q) = \sum_{i=0}^{N-1} I_1(g_i^Q) \qquad \text{and} \qquad \delta_N(g_N^Q) = \frac{1}{N}\sum_{i=0}^{N-1} \delta_1(g_i^Q), \tag{4.101}$$

where $g_i^Q = g_{U_i'|U_i}$ specifies the conditional pdf for the mapping of a vector component U_i onto its reconstruction U_i'. Consequently, the N-th order distortion rate function $D_N(R)$ can be expressed by

$$D_N(R) = \frac{1}{N}\sum_{i=0}^{N-1} D_i(R_i) \qquad \text{with} \qquad R = \frac{1}{N}\sum_{i=0}^{N-1} R_i, \tag{4.102}$$

where $R_i(D_i)$ denotes the first order rate distortion function for a vector component U_i. The first order distortion rate function for Gaussian sources has been derived in sec. 4.4.1 and is given by

$$D_i(R_i) = \sigma_i^2\, 2^{-2R_i}. \tag{4.103}$$

The variances σ_i^2 of the vector component U_i are equal to the eigenvalues $\xi_i^{(N)}$ of the N-th order autocovariance matrix $\boldsymbol{C}_N$. Hence, the N-th order distortion rate function can be written as

$$D_N(R) = \frac{1}{N}\sum_{i=0}^{N-1} \xi_i^{(N)}\, 2^{-2R_i} \qquad \text{with} \qquad R = \frac{1}{N}\sum_{i=0}^{N-1} R_i. \tag{4.104}$$

With the inequality of arithmetic and geometric means, which holds with equality if and only if all elements have the same value, we obtain

$$D_N(R) \geq \left(\prod_{i=0}^{N-1} \xi_i^{(N)} \, 2^{-2R_i} \right)^{\frac{1}{N}} = \left(\prod_{i=0}^{N-1} \xi_i^{(N)} \right)^{\frac{1}{N}} \cdot \; 2^{-2R} = \tilde{\xi}^{(N)} \cdot \; 2^{-2R}, \tag{4.105}$$

where $\tilde{\xi}^{(N)}$ denotes the geometric mean of the eigenvalues $\xi_i^{(N)}$. For a given N-th order mutual information R, the distortion is minimized if and only if $\xi_i^{(N)} 2^{-2R_i}$ is equal to $\tilde{\xi}^{(N)} 2^{-2R}$ for all $i = 0, \cdots, N-1$, which yields

$$R_i = R + \frac{1}{2} \log_2 \frac{\xi_i^{(N)}}{\tilde{\xi}^{(N)}}. \tag{4.106}$$

In the above result, we have ignored the fact that the mutual information R_i for a component U_i cannot be less than zero. Since the distortion rate function given in (4.103) is steeper at low R_i, the mutual information R_i for components with $\xi_i^{(N)} < \tilde{\xi}^{(N)} 2^{-2R}$ has to be set equal to zero and the mutual information R has to be distributed among the remaining components in order to minimize the distortion. This can be elegantly specified by introducing a parameter θ, with $\theta \geq 0$, and setting the component distortions according to

$$D_i = \min \left(\theta, \, \xi_i^{(N)} \right). \tag{4.107}$$

This concept is also know as inverse water-filling for independent Gaussian sources [57], where the parameter θ can be interpreted as the water level. Using (4.103), we obtain for the mutual information R_i,

$$R_i = \frac{1}{2} \log_2 \frac{\xi_i^{(N)}}{\min \left(\theta, \, \xi_i^{(N)} \right)} = \max \left(0, \, \frac{1}{2} \log_2 \frac{\xi_i^{(N)}}{\theta} \right). \tag{4.108}$$

The N-th order rate distortion function $R_N(D)$ can be expressed by the following parametric formulation, with $\theta \geq 0$,

$$D_N(\theta) \; = \; \frac{1}{N} \sum_{i=0}^{N-1} D_i = \frac{1}{N} \sum_{i=0}^{N-1} \min \left(\theta, \, \xi_i^{(N)} \right), \tag{4.109}$$

$$R_N(\theta) \; = \; \frac{1}{N} \sum_{i=0}^{N-1} R_i = \frac{1}{N} \sum_{i=0}^{N-1} \max \left(0, \, \frac{1}{2} \log_2 \frac{\xi_i^{(N)}}{\theta} \right). \tag{4.110}$$

The rate distortion function $R(D)$ for the stationary Gaussian random process $\{S_n\}$ is given by the limit

$$R(D) = \lim_{N\to\infty} R_N(D), \tag{4.111}$$

which yields the parametric formulation, with $\theta > 0$,

$$D(\theta) = \lim_{N\to\infty} D_N(\theta), \qquad R(\theta) = \lim_{N\to\infty} R_N(\theta). \tag{4.112}$$

For Gaussian processes with zero mean ($\boldsymbol{C}_N = \boldsymbol{R}_N$), we can apply the theorem for sequences of Toeplitz matrices (4.76) to express the rate distortion function using the power spectral density $\Phi_{SS}(\omega)$ of the source. A parametric formulation, with $\theta \geq 0$, for the rate distortion function $R(D)$ for a stationary Gaussian source with zero mean and a power spectral density $\Phi_{SS}(\omega)$ is given by

$$D(\theta) = \frac{1}{2\pi}\int_{-\pi}^{\pi} \min\left(\theta,\, \Phi_{SS}(\omega)\right)\, \mathrm{d}\omega, \tag{4.113}$$

$$R(\theta) = \frac{1}{2\pi}\int_{-\pi}^{\pi} \max\left(0,\, \frac{1}{2}\log_2 \frac{\Phi_{SS}(\omega)}{\theta}\right)\, \mathrm{d}\omega. \tag{4.114}$$

The minimization in the parametric formulation (4.113) and (4.114) of the rate distortion function is illustrated in Fig. 4.6. It can be interpreted that at each frequency, the variance of the corresponding frequency component as given by the power spectral density $\Phi_{SS}(\omega)$ is compared to the parameter θ, which represents the mean squared error of the frequency component. If $\Phi_{SS}(\omega)$ is found to be larger than θ, the mutual information is set equal to $\frac{1}{2}\log_2 \frac{\Phi_{SS}(\omega)}{\theta}$, otherwise a mutual information of zero is assigned to that frequency component.

For stationary zero-mean Gauss-Markov sources with a variance σ^2 and a correlation coefficient ρ, the power spectral density $\Phi_{SS}(\omega)$ is given by (4.78). If we choose the parameter θ according to

$$\theta \geq \min_{\forall\omega} \Phi_{SS}(\omega) = \sigma^2\,\frac{1-\rho^2}{1-2\rho+\rho^2} = \sigma^2\,\frac{1-\rho}{1+\rho}, \tag{4.115}$$

we obtain the parametric equations

$$D(\theta) = \frac{1}{2\pi}\int_{-\pi}^{\pi} \theta\, \mathrm{d}\omega = \theta, \tag{4.116}$$

$$R(\theta) = \frac{1}{4\pi}\int_{-\pi}^{\pi} \log_2 \frac{\Phi_{SS}(\omega)}{\theta}\, \mathrm{d}\omega = \frac{1}{2}\log_2 \frac{\sigma^2\,(1-\rho^2)}{\theta}, \tag{4.117}$$

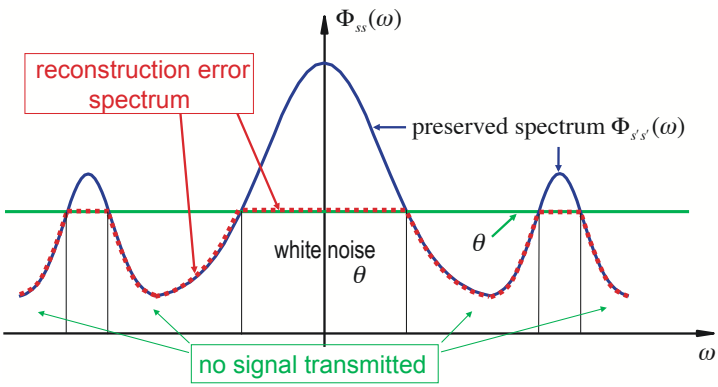


Fig. 4.6 Illustration of parametric equations for the rate distortion function of stationary Gaussian processes.

where we reused (4.79) for calculating the integral for $R(\theta)$. Since rate distortion functions are non-increasing, we can conclude that, for distortions less than or equal to $\sigma^2(1-\rho)/(1+\rho)$, the rate distortion function of a stationary Gauss-Markov process is equal to its Shannon lower bound,

$$R(D) = \frac{1}{2}\log_2 \frac{\sigma^2\,(1-\rho^2)}{D} \qquad \text{for} \quad D \le \sigma^2\,\frac{1-\rho}{1+\rho}. \tag{4.118}$$

Conversely, for rates $R \ge \log_2(1+\rho)$, the distortion rate function of a stationary Gauss-Markov process coincides with Shannon lower bound,

$$D(R) = (1-\rho)^2 \cdot \sigma^2 \cdot 2^{-2R} \qquad \text{for} \quad R \ge \log_2(1+\rho). \tag{4.119}$$

For Gaussian iid sources ($\rho = 0$), these results are identical to (4.87) and (4.88). Fig. 4.7 shows distortion rate functions for stationary Gauss-Markov processes with different correlation factors ρ. The distortion is plotted as signal-to-noise ratio $SNR = 10\log_{10}(\sigma^2/D)$.

We have noted above that the rate distortion function of the Gaussian iid process with a given variance specifies an upper bound for the rate distortion functions of all iid processes with the same variance. This statement can be generalized to stationary Gaussian processes with memory. The rate distortion function of the stationary zero-mean Gaussian process as given parametrically by (4.113) and (4.114) specifies an upper limit for the rate distortion functions of all other station-

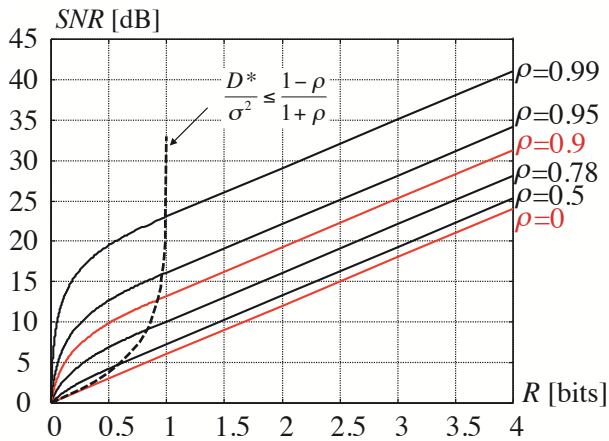


Fig. 4.7 Distortion rate functions for Gauss-Markov processes with different correlation factors ρ. The distortion D is plotted as signal-to-noise ratio $SNR = 10\log_{10}(\sigma^2/D)$.

ary processes with the same power spectral density $\Phi_{SS}(\omega)$. A proof of this statement can be found in [4].

4.5 Summary of Rate Distortion Theory

Rate distortion theory addresses the problem of finding the greatest lower bound for the average number of bits that is required for representing a signal without exceeding a given distortion. We introduced the operational rate distortion function that specifies this fundamental bound as infimum of over all source codes. A fundamental result of rate distortion theory is that the operational rate distortion function is equal to the information rate distortion function, which is defined as infimum over all conditional pdf's for the reconstructed samples given the original samples. Due to this equality, both the operational and the information rate distortion function are usually referred to as the rate distortion function. It has further been noted that, for the MSE distortion measure, the lossless coding theorem specifying that the average codeword length per symbol cannot be less than the entropy rate represents a special case of rate distortion theory for discrete sources with zero distortion.

For most sources and distortion measures, it is not known how to

analytically derive the rate distortion function. A useful lower bound for the rate distortion function is given by the so-called Shannon lower bound. The difference between the Shannon lower bound and the rate distortion function approaches zero as the distortion approaches zero or the rate approaches infinity. Due to this property, it represents a suitable reference for evaluating the performance of lossy coding schemes at high rates. For the MSE distortion measure, an analytical expression for the Shannon lower bound can be given for typical iid sources as well as for general stationary Gaussian sources.

An important class of processes is the class of stationary Gaussian processes. For Gaussian iid processes and MSE distortion, the rate distortion function coincides with the Shannon lower bound for all rates. The rate distortion function for general stationary Gaussian sources with zero mean and MSE distortion can be specified as a parametric expression using the power spectral density. It has also been noted that the rate distortion function of the stationary Gaussian process with zero mean and a particular power spectral density represents an upper bound for all stationary processes with the same power spectral density, which leads to the conclusion that Gaussian sources are the most difficult to code.

5

Quantization

Lossy source coding systems, which we have introduced in chapter 4, are characterized by the fact that the reconstructed signal is not identical to the source signal. The process that introduces the corresponding loss of information (or signal fidelity) is called *quantization*. An apparatus or algorithmic specification that performs the quantization process is referred to as *quantizer*. Each lossy source coding system includes a quantizer. The rate distortion point associated with a lossy source coding system is to a wide extent determined by the used quantization process. For this reason, the analysis of quantization techniques is of fundamental interest for the design of source coding systems.

In this chapter, we analyze the quantizer design and the performance of various quantization techniques with the emphasis on scalar quantization, since it is the most widely used quantization technique in video coding. To illustrated the inherent limitation of scalar quantization, we will also briefly introduce the concept of vector quantization and show its advantage with respect to the achievable rate distortion performance. For further details, the reader is referred to the comprehensive treatment of quantization in [16] and the overview of the history and theory of quantization in [28].

5.1 Structure and Performance of Quantizers

In the broadest sense, quantization is an irreversible deterministic mapping of an input quantity to an output quantity. For all cases of practical interest, the set of obtainable values for the output quantity is finite and includes fewer elements than the set of possible values for the input quantity. If the input quantity and the output quantity are scalars, the process of quantization is referred to as *scalar quantization*. A very simple variant of scalar quantization is the rounding of a real input value to its nearest integer value. Scalar quantization is by far the most popular form of quantization and is used in virtually all video coding applications. However, as we will see later, there is a gap between the operational rate distortion curve for optimal scalar quantizers and the fundamental rate distortion bound. This gap can only be reduced if a vector of more than one input sample is mapped to a corresponding vector of output samples. In this case, the input and output quantity are vectors and the quantization process is referred to as *vector quantization*. Vector quantization can asymptotically achieve the fundamental rate distortion bound if the number of samples in the input and output vector approaches infinity.

A quantizer Q of dimension N specifies a mapping of the N-dimensional Euclidean space $\mathcal{R}^N$ into a finite[1] set of reconstruction vectors inside the N-dimensional Euclidean space $\mathcal{R}^N$,

$$Q: \ \mathcal{R}^N \rightarrow \{\boldsymbol{s'}_0, \boldsymbol{s'}_1, \cdots, \boldsymbol{s'}_{K-1}\} . \tag{5.1}$$

If the dimension N of the quantizer Q is equal to 1, it is a scalar quantizer; otherwise, it is a vector quantizer. The number K of reconstruction vectors is also referred to as the size of the quantizer Q. The deterministic mapping Q associates a subset $\mathcal{C}_i$ of the N-dimensional Euclidean space $\mathcal{R}^N$ with each of the reconstruction vectors $\boldsymbol{s'}_i$. The subsets $\mathcal{C}_i$, with $0 \leq i < K$, are called *quantization cells* and are defined by

$$\mathcal{C}_i = \left\{\boldsymbol{s} \in \mathcal{R}^N : \ Q(\boldsymbol{s}) = \boldsymbol{s'}_i\right\} . \tag{5.2}$$

[1] Although we restrict our considerations to finite sets of reconstruction vectors, some of the presented quantization methods and derivations are also valid for countably infinite sets of reconstruction vectors.

From this definition, it follows that the quantization cells $\mathcal{C}_i$ form a partition of the N-dimensional Euclidean space $\mathcal{R}^N$,

$$\bigcup_{i=0}^{K-1} \mathcal{C}_i = \mathcal{R}^N \qquad \text{with} \qquad \forall i \neq j: \;\; \mathcal{C}_i \cap \mathcal{C}_j = \emptyset. \tag{5.3}$$

Given the quantization cells $\mathcal{C}_i$ and the associated reconstruction values $\boldsymbol{s}'_i$, the quantization mapping Q can be specified by

$$Q(\boldsymbol{s}) = \boldsymbol{s}'_i \qquad \forall \boldsymbol{s} \in \mathcal{C}_i. \tag{5.4}$$

A quantizer is completely specified by the set of reconstruction values and the associated quantization cells.

For analyzing the design and performance of quantizers, we consider the quantization of symbol sequences $\{s_n\}$ that represent realizations of a random process $\{S_n\}$. For the case of vector quantization ($N > 1$), the samples of the input sequence $\{s_n\}$ shall be arranged in vectors, resulting in a sequence of symbol vectors $\{\boldsymbol{s}_n\}$. Usually, the input sequence $\{s_n\}$ is decomposed into blocks of N samples and the components of an input vector $\boldsymbol{s}_n$ are build be the samples of such a block, but other arrangements are also possible. In any case, the sequence of input vectors $\{\boldsymbol{s}_n\}$ can be considered to represent a realization of a vector random process $\{\boldsymbol{S}_n\}$. It should be noted that the domain of the input vectors $\boldsymbol{s}_n$ can be a subset of the N-dimensional space $\mathcal{R}^N$, which is the case if the random process $\{S_n\}$ is discrete or its marginal pdf $f(\boldsymbol{s})$ is zero outside a finite interval. However, even in this case, we can generally consider quantization as a mapping of the N-dimensional Euclidean space $\mathcal{R}^N$ into a finite set of reconstruction vectors.

Fig. 5.1 shows a block diagram of a quantizer Q. Each input vector $\boldsymbol{s}_n$ is mapped onto one of the reconstruction vectors, given by $Q(\boldsymbol{s}_n)$.

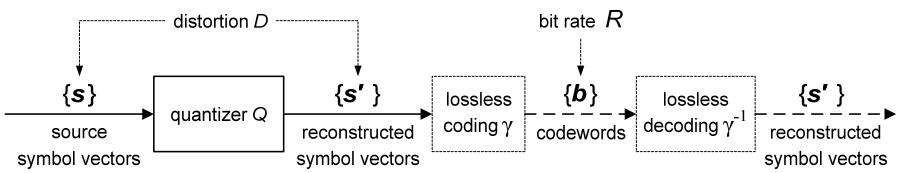


Fig. 5.1 Basic structure of a quantizer Q in combination with a lossless coding γ.

The average distortion D per sample between the input and output vectors depends only on the statistical properties of the input sequence $\{\boldsymbol{s}_n\}$ and the quantization mapping Q. If the random process $\{\boldsymbol{S}_n\}$ is stationary, it can be expressed by

$$D = E\{d_N(\boldsymbol{S}_n, Q(\boldsymbol{S}_n))\} = \frac{1}{N} \sum_{i=0}^{K-1} \int_{\mathcal{C}_i} d_N(\boldsymbol{s}, Q(\boldsymbol{s}))\ f_{\boldsymbol{S}}(\boldsymbol{s})\, \mathrm{d}\boldsymbol{s}, \tag{5.5}$$

where $f_{\boldsymbol{S}}$ denotes the joint pdf of the vector components of the random vectors $\boldsymbol{S}_n$. For the MSE distortion measure, we obtain

$$D = \frac{1}{N} \sum_{i=0}^{K-1} \int_{\mathcal{C}_i} f_{\boldsymbol{S}}(\boldsymbol{s})\, (\boldsymbol{s} - \boldsymbol{s'}_i)^T (\boldsymbol{s} - \boldsymbol{s'}_i)\ \mathrm{d}\boldsymbol{s}. \tag{5.6}$$

Unlike the distortion D, the average transmission rate is not only determined by the quantizer Q and the input process. As illustrated in Fig. 5.1, we have to consider the lossless coding γ by which the sequence of reconstruction vectors $\{Q(\boldsymbol{s}_n)\}$ is mapped onto a sequence of codewords. For calculating the performance of a quantizer or for designing a quantizer we have to make reasonable assumptions about the lossless coding γ. It is certainly not a good idea to assume a lossless coding with an average codeword length per symbol close to the entropy for the design, but to use the quantizer in combination with fixed-length codewords for the reconstruction vectors. Similarly, a quantizer that has been optimized under the assumption of fixed-length codewords is not optimal if it is used in combination with advanced lossless coding techniques such as Huffman coding or arithmetic coding.

The rate R of a coding system consisting of a quantizer Q and a lossless coding γ is defined as the average codeword length per input sample. For stationary input processes $\{\boldsymbol{S}_n\}$, it can be expressed by

$$R = \frac{1}{N} E\{\, |\gamma(\, Q(\boldsymbol{S}_n)\,)|\,\} = \frac{1}{N} \sum_{i=0}^{N-1} p(\boldsymbol{s'}_i) \cdot |\gamma(\boldsymbol{s'}_i)|\,, \tag{5.7}$$

where $|\gamma(\boldsymbol{s'}_i)|$ denotes the average codeword length that is obtained for a reconstruction vector $\boldsymbol{s'}_i$ with the lossless coding γ and $p(\boldsymbol{s'}_i)$ denotes the pmf for the reconstruction vectors, which is given by

$$p(\boldsymbol{s'}_i) = \int_{\mathcal{C}_i} f_{\boldsymbol{S}}(\boldsymbol{s})\ \mathrm{d}\boldsymbol{s}. \tag{5.8}$$

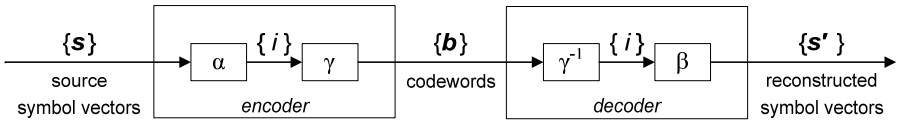


Fig. 5.2 Lossy source coding system consisting of a quantizer, which is decomposed into an encoder mapping α and a decoder mapping β, and a lossless coder γ.

The probability of a reconstruction vector does not depend on the reconstruction vector itself, but only on the associated quantization cell $\mathcal{C}_i$.

A quantizer Q can be decomposed into two parts, an encoder mapping α which maps the input vectors $\boldsymbol{s}_n$ to quantization indexes i, with $0 \leq i < K$, and a decoder mapping β which maps the quantization indexes i to the associated reconstruction vectors $\boldsymbol{s}'_i$. The quantizer mapping can then be expressed by $Q(\boldsymbol{s}) = \alpha(\beta(\boldsymbol{s}))$. The loss of signal fidelity is introduced as a result of the encoder mapping α, the decoder mapping β merely maps the quantization indexes i to the associated reconstruction vectors $\boldsymbol{s}'_i$. The combination of the encoder mapping α and the lossless coding γ forms an encoder of a lossy source coding system as illustrated in Fig. 5.2. The corresponding decoder is given by the inverse lossless coding γ^{-1} and the decoder mapping β.

5.2 Scalar Quantization

In scalar quantization ($N = 1$), the input and output quantity are scalars. Hence, a scalar quantizer Q of size K specifies a mapping of the real line $\mathcal{R}$ into a set of K reconstruction levels,

$$Q: \ \mathcal{R} \rightarrow \{s'_0, s'_1, \cdots, s'_{K-1}\}. \tag{5.9}$$

In the general case, a quantization cell $\mathcal{C}_i$ corresponds to a set of intervals of the real line. We restrict our considerations to *regular scalar quantizers* for which each quantization cell $\mathcal{C}_i$ represents a single interval of the real line $\mathcal{R}$ and the reconstruction levels s'_i are located inside the associated quantization cells $\mathcal{C}_i$. Without loss of generality, we further assume that the quantization cells are ordered in increasing order of the values of their lower interval boundary. When we further assume that the quantization intervals include the lower, but not the

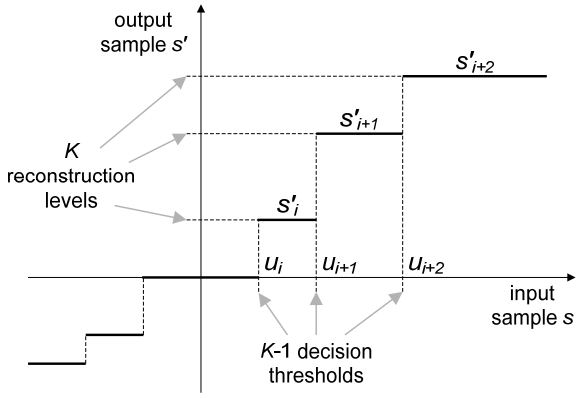


Fig. 5.3 Input-output function Q of a scalar quantizer.

higher interval boundary, each quantization cell can be represented by a half-open[2] interval $\mathcal{C}_i = [u_i, u_{i+1})$. The interval boundaries u_i are also referred to as *decision thresholds*. The interval sizes $\Delta_i = u_{i+1} - u_i$ are called *quantization step sizes*. Since the quantization cells must form a partition of the real line $\mathcal{R}$, the values u_0 and u_K are fixed and given by $u_0 = -\infty$ and $u_K = \infty$. Consequently, K reconstruction levels and $K-1$ decision thresholds can be chosen in the quantizer design.

The quantizer mapping Q of a scalar quantizer as defined above can be represented by a piecewise-constant input-output function as illustrated in Fig. 5.3. All input values s with $u_i \leq s < u_{i+1}$ are assigned to the corresponding reproduction level s'_i.

In the following treatment of scalar quantization, we generally assume that the input process is stationary. For continuous random processes, scalar quantization can then can be interpreted as a discretization of the marginal pdf $f(s)$ as illustrated in Fig. 5.4.

For any stationary process $\{S\}$ with a marginal pdf $f(s)$, the quantizer output is a discrete random process $\{S'\}$ with a marginal pmf

$$p(s'_i) = \int_{u_i}^{u_{i+1}} f(s)\, \mathrm{d}s. \tag{5.10}$$

[2] In strict mathematical sense, the first quantization cell is an open interval $\mathcal{C}_0 = (-\infty, u_1)$.

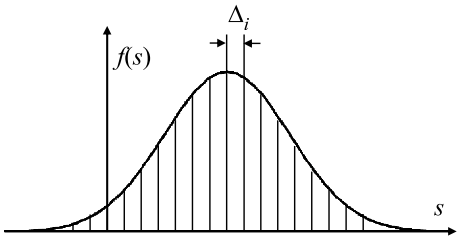


Fig. 5.4 Scalar quantization as discretization of the marginal pdf $f(s)$.

The average distortion D (for the MSE distortion measure) is given by

$$D = E\{d(S, Q(S))\} = \sum_{i=0}^{K-1} \int_{u_i}^{u_{i+1}} (s - s'_i)^2 \cdot f(s)\, \mathrm{d}s. \tag{5.11}$$

The average rate R depends on the lossless coding γ and is given by

$$R = E\{|\gamma(Q(S))|\} = \sum_{i=0}^{N-1} p(s'_i) \cdot |\gamma(s'_i)|. \tag{5.12}$$

5.2.1 Scalar Quantization with Fixed-Length Codes

We will first investigate scalar quantizers in connection with fixed-length codes. The lossless coding γ is assumed to assign a codeword of the same length to each reconstruction level. For a quantizer of size K, the codeword length must be greater than or equal to $\lceil \log_2 K \rceil$. Under these assumptions, the quantizer size K should be a power of 2. If K is not a power of 2, the quantizer requires the same minimum codeword length as a quantizer of size $K' = 2^{\lceil \log_2 K \rceil}$, but since $K < K'$, the quantizer of size K' can achieve a smaller distortion. For simplifying the following discussion, we define the rate R according to

$$R = \log_2 K, \tag{5.13}$$

but inherently assume that K represents a power of 2.

Pulse-Code-Modulation (PCM). A very simple form of quantization is the *pulse-code-modulation* (PCM) for random processes with a finite amplitude range. PCM is a quantization process for which all

quantization intervals have the same size Δ and the reproduction values s'_i are placed in the middle between the decision thresholds u_i and u_{i+1}. For general input signals, this is not possible since it results in an infinite number of quantization intervals K and hence an infinite rate for our fixed-length code assumption. However, if the input random process has a finite amplitude range of $[s_{\min}, s_{\max}]$, the quantization process is actually a mapping of the finite interval $[s_{\min}, s_{\max}]$ to the set of reproduction levels. Hence, we can set $u_0 = s_{\min}$ and $u_K = s_{\max}$. The width $A = s_{\max} - s_{\min}$ of the amplitude interval is then evenly split into K quantization intervals, resulting in a quantization step size

$$\Delta = \frac{A}{K} = A \cdot 2^{-R}. \tag{5.14}$$

The quantization mapping for PCM can be specified by

$$Q(s) = \left\lfloor \frac{s - s_{\min}}{\Delta} + 0.5 \right\rfloor \cdot \Delta + s_{\min}. \tag{5.15}$$

As an example, we consider PCM quantization of a stationary random process with an uniform distribution, $f(s) = \frac{1}{A}$ for $-\frac{A}{2} \leq s \leq \frac{A}{2}$. The distortion as defined in (5.11) becomes

$$D = \sum_{i=0}^{K-1} \int_{s_{\min}+i\Delta}^{s_{\min}+(i+1)\Delta} \frac{1}{A} \left(s - s_{\min} - \left(i + \frac{1}{2} \right) \cdot \Delta \right)^2 \, \mathrm{d}s. \tag{5.16}$$

By carrying out the integration and inserting (5.14), we obtain the operational distortion rate function,

$$D_{\mathrm{PCM,uniform}}(R) = \frac{A^2}{12} \cdot 2^{-2R} = \sigma^2 \cdot 2^{-2R}. \tag{5.17}$$

For stationary random processes with an infinite amplitude range, we have to choose $u_0 = -\infty$ and $u_K = \infty$. The inner interval boundaries u_i, with $0 < i < K$, and the reconstruction levels s'_i can be evenly distributed around the mean value μ of the random variables S. For symmetric distributions ($\mu = 0$), this gives

$$s'_i = \left(i - \frac{K-1}{2} \right) \cdot \Delta \qquad \text{for} \qquad 0 \leq i < K, \tag{5.18}$$

$$u_i = \left(i - \frac{K}{2} \right) \cdot \Delta \qquad \text{for} \qquad 0 < i < K. \tag{5.19}$$

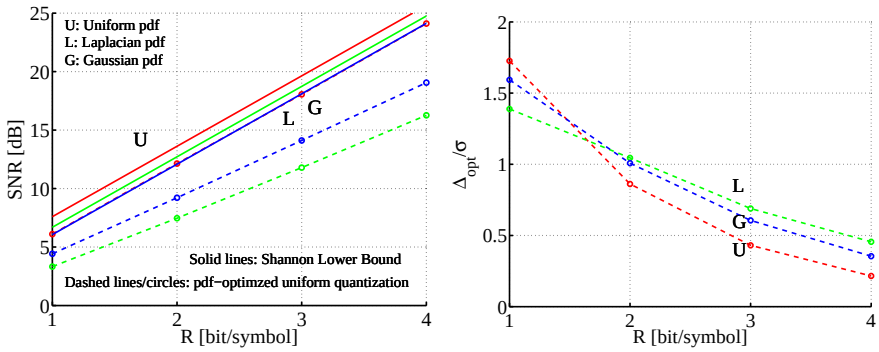


Fig. 5.5 PCM quantization of stationary random processes with uniform (U), Laplacian (L), and Gaussian (G) distributions: (left) operational distortion rate functions in comparison to the corresponding Shannon lower bounds (for variances $\sigma^2 = 1$); (right) optimal quantization step sizes.

Inserting these expressions into (5.11) yields an expression for the distortion $D(\Delta)$ that depends only on the quantization step size Δ for a given quantizer size K. The quantization step size Δ can be chosen in a way that the distortion is minimized. As an example, we minimized the distortions for the uniform, Laplacian, and Gaussian distribution for given quantizer sizes K by numerical optimization. The obtained operational rate distortion curves and corresponding quantization step sizes are depicted in Fig. 5.5. The numerically obtained results for the uniform distribution are consistent with (5.17) and (5.14). For the Laplacian and Gaussian distribution, the loss in SNR with respect to the Shannon lower bound (high-rate approximation of the distortion rate function) is significant and increases toward higher rates.

Pdf-Optimized Scalar Quantization with Fixed-Length Codes. For the application of PCM quantization to stationary random processes with an infinite amplitude interval, we have chosen the quantization step size for a given quantizer size K by minimizing the distortion. A natural extension of this concept is to minimize the distortion with respect to all parameters of a scalar quantizer of a given size K. The optimization variables are the $K-1$ decision thresholds u_i, with $0 < i < K$, and the K reconstruction levels s'_i, with

$0 \leq i < K$. The obtained quantizer is called a *pdf-optimized scalar quantizer with fixed-length codes.*

For deriving a condition for the reconstruction levels s'_i, we first assume that the decision thresholds u_i are given. The overall distortion (5.11) is the sum of the distortions D_i for the quantization intervals $\mathcal{C}_i = [u_i, u_{u+1})$. For given decision thresholds, the interval distortions D_i are mutually independent and are determined by the corresponding reconstruction levels s'_i,

$$D_i(s'_i) = \int_{u_i}^{u_{i+1}} d_1(s, s'_i) \cdot f(s) \, \mathrm{d}s. \tag{5.20}$$

By using the conditional distribution $f(s|s'_i) = f(s) \cdot p(s'_i)$, we obtain

$$D_i(s'_i) = \frac{1}{p(s'_i)} \int\limits_{u_i}^{u_{i+1}} d_1(s, s'_i) \cdot f(s|s'_i) \, \mathrm{d}s = \frac{E\{d_1(S, s'_i) | \, S \in \mathcal{C}_i\}}{p(s'_i)}. \tag{5.21}$$

Since $p(s'_i)$ does not depend on s'_i, the optimal reconstruction levels s'^*_i are given by

$$s'^*_i = \arg\min_{s' \in \mathcal{R}} \; E\big\{d_1(S, s') | \, S \in \mathcal{C}_i\big\}, \tag{5.22}$$

which is also called the *generalized centroid condition.* For the squared error distortion measure $d_1(s, s') = (s - s')^2$, the optimal reconstruction levels s'^*_i are the conditional means (centroids)

$$s'^*_i = E\{S | \, S \in \mathcal{C}_i\} = \frac{\int_{u_i}^{u_{i+1}} s \cdot f(s) \, \mathrm{d}s}{\int_{u_i}^{u_{i+1}} f(s) \, \mathrm{d}s}. \tag{5.23}$$

This can be easily proved by the inequality

$$\begin{aligned} E\big\{(S - s'_i)^2\big\} &= E\big\{(S - E\{S\} + E\{S\} - s'_i)^2\big\} \\ &= E\big\{(S - E\{S\})^2\big\} + (E\{S\} - s'_i)^2 \\ &\geq E\big\{(S - E\{S\})^2\big\}. \end{aligned} \tag{5.24}$$

If the reproduction levels s'_i are given, the overall distortion D is minimized if each input value s is mapped to the reproduction level s'_i that minimizes the corresponding sample distortion $d_1(s, s'_i)$,

$$Q(s) = \arg\min_{\forall s'_i} \; d_1(s, s'_i). \tag{5.25}$$

This condition is also referred to as the *nearest neighbor condition*. Since a decision threshold u_i influences only the distortions D_i of the neighboring intervals, the overall distortion is minimized if

$$d_1(u_i, s'_{i-1}) = d_1(u_i, s'_i) \tag{5.26}$$

holds for all decision thresholds u_i, with $0 < i < K$. For the squared error distortion measure, the optimal decision thresholds u_i^*, with $0 < i < K$, are given by

$$u_i^* = \frac{1}{2}(s'_{i-1} + s'_i). \tag{5.27}$$

The expressions (5.23) and (5.27) can also be obtained by setting the partial derivatives of the distortion (5.11) with respect to the decision thresholds u_i and the reconstruction levels s'_i equal to zero [56].

The Lloyd Algorithm. The necessary conditions for the optimal reconstruction levels (5.22) and decision thresholds (5.25) depend on each other. A corresponding iterative algorithm for minimizing the distortion of a quantizer of given size K was suggested by Lloyd [49] and is commonly called the *Lloyd algorithm*. The obtained quantizer is referred to as *Lloyd quantizer* or *Lloyd-Max*[3] *quantizer*. For a given pdf $f(s)$, first an initial set of unique reconstruction levels $\{s'_i\}$ is arbitrarily chosen, then the decision thresholds $\{u_i\}$ and reconstruction levels $\{s'_i\}$ are alternately determined according to (5.25) and (5.22), respectively, until the algorithm converges. It should be noted that the fulfillment of the conditions (5.22) and (5.25) is in general not sufficient to guarantee the optimality of the quantizer. The conditions are only sufficient if the pdf $f(s)$ is log-concave. One of the examples, for which the Lloyd algorithm yields a unique solution independent of the initial set of reconstruction levels, is the Gaussian pdf.

Often, the marginal pdf $f(s)$ of a random process is not known a priori. In such a case, the Lloyd algorithm can be applied using a training set. If the training set includes a sufficiently large number of samples, the obtained quantizer is an accurate approximation of the

[3] Lloyd and Max independently observed the two necessary conditions for optimality.

Lloyd quantizer. Using the encoder mapping α (see sec. 5.1), the Lloyd algorithm for a training set of samples $\{s_n\}$ and a given quantizer size K can be stated as follows:

(1) Choose an initial set of unique reconstruction levels $\{s'_i\}$.

(2) Associate all samples of the training set $\{s_n\}$ with one of the quantization intervals $\mathcal{C}_i$ according to

$$\alpha(s_n) = \arg\min_{\forall i} \; d_1(s_n, s'_i) \qquad \text{(nearest neighbor condition)}$$

and update the decision thresholds $\{u_i\}$ accordingly.

(3) Update the reconstruction levels $\{s'_i\}$ according to

$$s'_i = \arg\min_{s' \in \mathcal{R}} \; E\big\{d_1(S, s') \,|\, \alpha(S) = i\big\}\,, \qquad \text{(centroid condition)}$$

where the expectation value is taken over the training set.

(4) Repeat the previous two steps until convergence.

Examples for the Lloyd Algorithm. As a first example, we applied the Lloyd algorithm with a training set of more than 10000 samples and the MSE distortion measure to a Gaussian pdf with unit variance. We used two different initializations for the reconstruction levels. Convergence was determined if the relative distortion reduction between two iterations steps was less than 1%, $(D_k - D_{k+1})/D_{k+1} < 0.01$. The algorithm quickly converged after 6 iterations for both initializations to the same overall distortion D_F^*. The obtained reconstruction levels $\{s'_i\}$ and decision thresholds $\{u_i\}$ as well as the iteration processes for the two initializations are illustrated in Fig. 5.6.

The same algorithm with the same two initializations was also applied to a Laplacian pdf with unit variance. Also for this distribution, the algorithm quickly converged after 6 iterations for both initializations to the same overall distortion D_F^*. The obtained quantizer and the iteration processes are illustrated in Fig. 5.7.

5.2.2 Scalar Quantization with Variable-Length Codes

We have investigated the design of quantizers that minimize the distortion for a given number K of reconstruction levels, which is equivalent

Decision thresholds u_i:

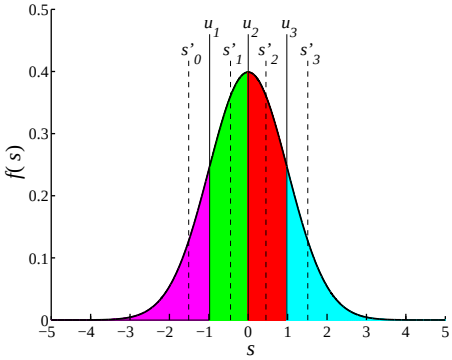
$$u_1 = -0.98, \quad u_2 = 0, \quad u_3 = 0.98$$

Decoding symbols s'_i:

$$s'_0 = -1.51, \quad s'_1 = -0.45$$

$$s'_2 = 0.45, \quad s'_3 = 1.51$$

Distortion: $D^*_F = 0.12 = 9.3$ dB



Initialization A:
$s'_i = -3.75 + 2.5 \cdot i$

Initialization B:
$s'_{3/0} = +/- 1.15, \; s'_{2/1} = +/- 0.32$

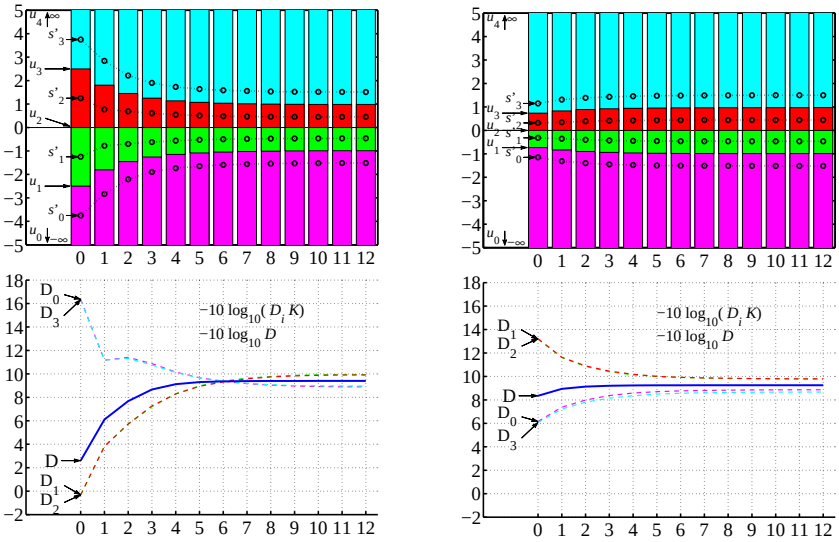


Fig. 5.6 Lloyd algorithm for a Gaussian pdf with unit variance and two initializations: (top) final reconstruction levels and decision thresholds; (middle) reconstruction levels and decision thresholds as function of the iteration step; (bottom) overall SNR and SNR for the quantization intervals as function of the iteration step.

to a quantizer optimization using the assumption that all reconstruction levels are signaled with codewords of the same length. Now we consider the quantizer design in combination with variable-length codes γ.

The average codeword length that is associated with a particular reconstruction level s'_i is denoted by $\bar{\ell}(s'_i) = |\gamma(s'_i)|$. If we use a scalar Huffman code, $\bar{\ell}(s'_i)$ is equal to the length of the codeword that is

Decision thresholds u_i:

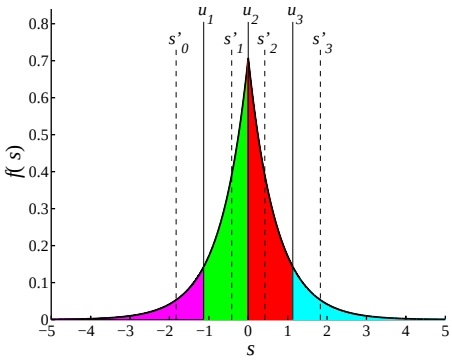
$$u_1 = -1.13, \quad u_2 = 0, \quad u_3 = 1.13$$

Decoding symbols s'_i:

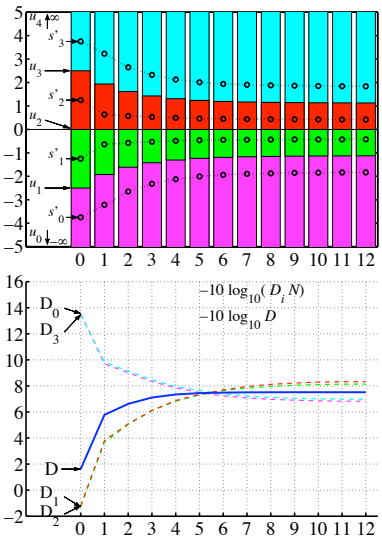
$$s'_0 = -1.83, \quad s'_1 = -0.42$$

$$s'_2 = 0.42, \quad s'_3 = 1.83$$

Distortion: $D_F^* = 0.18 = 7.55$ dB



Initialization A:
$s'_i = -3.75 + 2.5 \cdot i$



Initialization B:
$s'_{3/0} = +/- 1.15, \; s'_{2/1} = +/- 0.32$

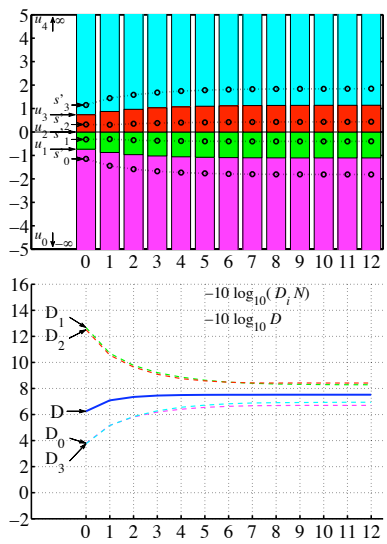


Fig. 5.7 Lloyd algorithm for a Laplacian pdf with unit variance and two initializations: (top) final reconstruction levels and decision thresholds; (middle) reconstruction levels and decision thresholds as function of the iteration step; (bottom) overall SNR and SNR for the quantization intervals as function of the iteration step.

assigned to s'_i. According to (5.12), the average rate R is given by

$$R = \sum_{i=0}^{N-1} p(s'_i) \cdot \bar{\ell}(s'_i). \tag{5.28}$$

The average distortion is the same as for scalar quantization with fixed-length codes and is given by (5.11).

Rate-Constrained Scalar Quantization. Since distortion and rate influence each other, they cannot be minimized independently. The optimization problem can be stated as

$$\min D \qquad \text{subject to} \qquad R \leq R_{\max}, \tag{5.29}$$

or equivalently,

$$\min R \qquad \text{subject to} \qquad D \leq D_{\max}, \tag{5.30}$$

with $R_{\max}$ and $D_{\max}$ being a given maximum rate and a maximum distortion, respectively. The constraint minimization problem can be formulated as unconstrained minimization of the Lagrangian functional

$$J = D + \lambda\, R = E\{d_1(S, Q(S))\} + \lambda\, E\big\{\bar{\ell}(Q(S))\big\}\,. \tag{5.31}$$

The parameter λ, with $0 \leq \lambda < \infty$, is referred to as Lagrange parameter. The solution of the minimization of (5.31) is a solution of the constrained minimization problems (5.29) and (5.30) in the following sense: If there is a Lagrangian parameter λ that yields a particular rate $R_{\max}$ (or particular distortion $D_{\max}$), the corresponding distortion D (or rate R) is a solution of the constraint optimization problem.

In order to derive necessary conditions similarly as for the quantizer design with fixed-length codes, we first assume that the decision thresholds u_i are given. Since the rate R is independent of the reconstruction levels s'_i, the optimal reconstruction levels are found by minimizing the distortion D. This is the same optimization problem as for the scalar quantizer with fixed-length codes. Hence, the optimal reconstruction levels s'^*_i are given by the generalized centroid condition (5.22).

The optimal average codeword lengths $\bar{\ell}(s'_i)$ also depend only on the decision thresholds u_i. Given the decision thresholds and thus the probabilities $p(s'_i)$, the average codeword lengths $\bar{\ell}(s'_i)$ can be determined. If we, for example, assume that the reconstruction levels are coded using a scalar Huffman code, the Huffman code could be constructed given the pmf $p(s'_i)$, which directly yields the codeword length $\bar{\ell}(s'_i)$. In general, it is however justified to approximate the average rate R by the entropy $H(S)$ and set the average codeword length equal to

$$\bar{\ell}(s'_i) = -\log_2 p(s'_i). \tag{5.32}$$

This underestimates the true rate by a small amount. For Huffman coding the difference is always less than 1 bit per symbol and for arithmetic coding it is usually much smaller. When using the entropy as approximation for the rate during the quantizer design, the obtained quantizer is also called an *entropy-constrained scalar quantizer*. At this point, we ignore that, for sources with memory, the lossless coding γ can employ dependencies between output samples, for example, by using block Huffman coding or arithmetic coding with conditional probabilities. This extension is discussed in sec. 5.2.6.

For deriving a necessary condition for the decision thresholds u_i, we now assume that the reconstruction levels s'_i and the average codeword length $\bar{\ell}(s'_i)$ are given. Similarly as for the nearest neighbor condition in sec. 5.2.1, the quantization mapping $Q(s)$ that minimizes the Lagrangian functional J is given by

$$Q(s) = \arg\min_{\forall s'_i} \; d_1(s, s'_i) + \lambda\, \bar{\ell}(s'_i). \tag{5.33}$$

A mapping $Q(s)$ that that minimizes the term $d(s, s'_i) + \lambda\bar{\ell}(s'_i)$ for each source symbol s minimizes also the expected value in (5.31). A rigorous proof of this statement can be found in [71]. The decision thresholds u_i have to be selected in a way that the term $d(s, s'_i) + \lambda\bar{\ell}(s'_i)$ is the same for both neighboring intervals,

$$d_1(u_i, s'_{i-1}) + \lambda\, \bar{\ell}(s'_{i-1}) = d_1(u_i, s'_i) + \lambda\, \bar{\ell}(s'_i). \tag{5.34}$$

For the MSE distortion measure, we obtain

$$u_i^* = \frac{1}{2}(s'_i + s'_{i+1}) + \frac{\lambda}{2} \cdot \frac{\bar{\ell}(s'_{i+1}) - \bar{\ell}(s'_i)}{s'_{i+1} - s'_i}. \tag{5.35}$$

The consequence is a shift of the decision threshold u_i from the midpoint between the reconstruction levels toward the interval with the longer average codeword length, i.e., the less probable interval.

Lagrangian Minimization. Lagrangian minimization as in (5.33) is a very important concept in modern video coding. Hence, we have conducted a simple experiment to illustrate the minimization approach. For that, we simulated the encoding of a 5-symbol sequence $\{s_i\}$. The

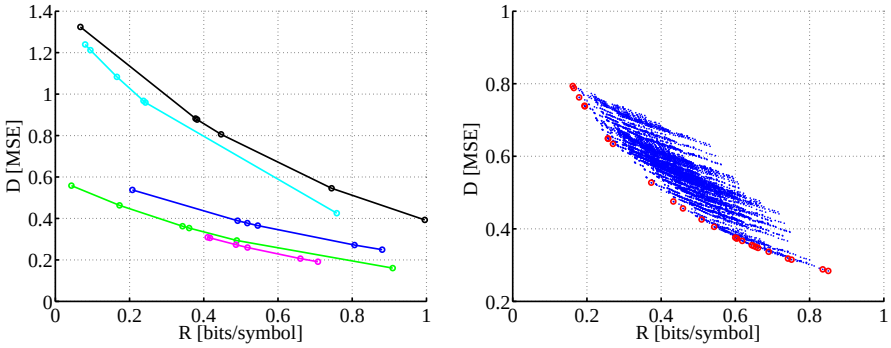


Fig. 5.8 Lagrangian minimization: (left) independent operational distortion rate curves for the 5 symbols, where each circle represents one of 6 available distortion rate points; (right) the small dots show the average distortion and rate for all possible combinations of the 5 different quantizers with their 6 rate distortion points, the circles show the solutions to the Lagrangian minimization problem.

symbols are assumed to be mutually independent and have different distributions. We have generated one operational distortion rate function $D_i(R) = a_i^2\ 2^{-2R}$ for each symbol, with a_i^2 being randomly chosen. For each operational distortion rate function we have selected 6 rate points $R_{i,k}$, which represent the available quantizers.

The Lagrangian optimization process is illustrated in Fig. 5.8. The diagram on the left shows the 5 operational distortion rate functions $D_i(R)$ with the available rate points $R_{i,k}$. The right diagram shows the average distortion and rate for each combination of rate points for encoding the 5-symbol sequence. The results of the minimization of $D_i(R_{i,k}) + \lambda R_{i,k}$ with respect to $R_{i,k}$ for different values of the Lagrange parameter λ are marked by circles. This experiment illustrates that the Lagrangian minimization approach yields a result on the convex hull of the admissible distortion rate points.

The Entropy-Constrained Lloyd Algorithm. Given the necessary conditions for an optimal quantizer with variable-length codes, we can construct an iterative design algorithm similar to the Lloyd algorithm. If we use the entropy as measure for the average rate, the algorithm is also referred to as *entropy-constrained Lloyd algorithm*. Using the encoder mapping α, the variant that uses a sufficiently large

training set $\{s_n\}$ can be stated as follows for a given value of λ:

(1) Choose an initial quantizer size N, an initial set of reconstruction levels $\{s'_i\}$, and an initial set of average codeword lengths $\bar{\ell}(s'_i)$.

(2) Associate all samples of the training set $\{s_n\}$ with one of the quantization intervals $\mathcal{C}_i$ according to

$$\alpha(s_n) = \arg\min_{\forall i} \; d_1(s_n, s'_i) + \lambda\, \bar{\ell}(s'_i)$$

and update the decision thresholds $\{u_i\}$ accordingly.

(3) Update the reconstruction levels $\{s'_i\}$ according to

$$s'_i = \arg\min_{s' \in \mathcal{R}} \; E\big\{d_1(S, s') \,|\, \alpha(S) = i\big\}\,,$$

where the expectation value is taken over the training set.

(4) Update the average codeword length $\bar{\ell}(s'_i)$ according to[4]

$$\bar{\ell}(s'_i) = -\log_2 p(s'_i).$$

(5) Repeat the previous three steps until convergence.

As mentioned above, the entropy constraint in the algorithm causes a shift of the cost function depending on the pmf $p(s'_i)$. If two decoding symbols s'_i and s'_{i+1} are competing, the symbol with larger popularity has higher chance of being chosen. The probability of a reconstruction level that is rarely chosen is further reduced. As a consequence, symbols get "removed" and the quantizer size K of the final result can be smaller than the initial quantizer size N.

The number N of initial reconstruction levels is critical to quantizer performance after convergence. Fig. 5.9 illustrates the result of the entropy-constrained Lloyd algorithm after convergence for a Laplacian pdf and different numbers of initial reconstruction levels, where the rate is measured as the entropy of the reconstruction symbols. It can be seen that a larger number of initial reconstruction levels always leads to a smaller or equal distortion (higher or equal SNR) at the same rate than a smaller number of initial reconstruction levels.

[4] In a variation of the entropy-constrained Lloyd algorithm, the average codeword lengths $\bar{\ell}(s'_i)$ can be determined by constructing a lossless code γ given the pmf $p(s'_i)$.

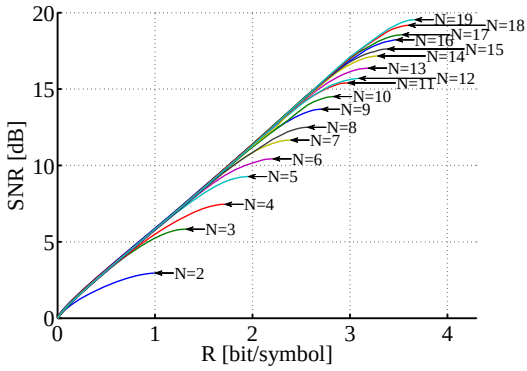


Fig. 5.9 Operational distortion rate curves after convergence of the entropy-constrained Lloyd algorithm for different numbers of initialized reconstruction levels. The rate R is measured as the entropy of the reconstruction symbols.

Examples for the Entropy-Constrained Lloyd Algorithm.
As a first example, we applied the entropy-constrained Lloyd algorithm with the MSE distortion to a Gaussian pdf with unit variance. The resulting average distortion D_F^* is 10.45 dB for an average rate R, measured as entropy, of 2 bit per symbol. The obtained optimal reconstruction levels and decision thresholds are depicted in Fig. 5.10. This figure also illustrates the iteration process for two different initializations. For initialization A, the initial number of reconstruction levels is sufficiently large and during the iteration process the size of the quantizer is reduced. With initialization B, however, the desired quantizer performance is not achieved, because the number of initial reconstruction levels is too small for the chosen value of λ.

The same experiment was done for a Laplacian pdf with unit variance. Here, the resulting average distortion D_F^* is 11.46 dB for an average rate R, measured as entropy, of 2 bit per symbol. The obtained optimal reconstruction levels and decision thresholds as well as the iteration processes are illustrated in Fig. 5.11. Similarly as for the Gaussian pdf, the number of initial reconstruction levels for the initialization B is too small for the chosen value of λ, so that the desired quantization performance is not achieved. For initialization A, the initial quantizer size is large enough and the number of quantization intervals is reduced during the iteration process.

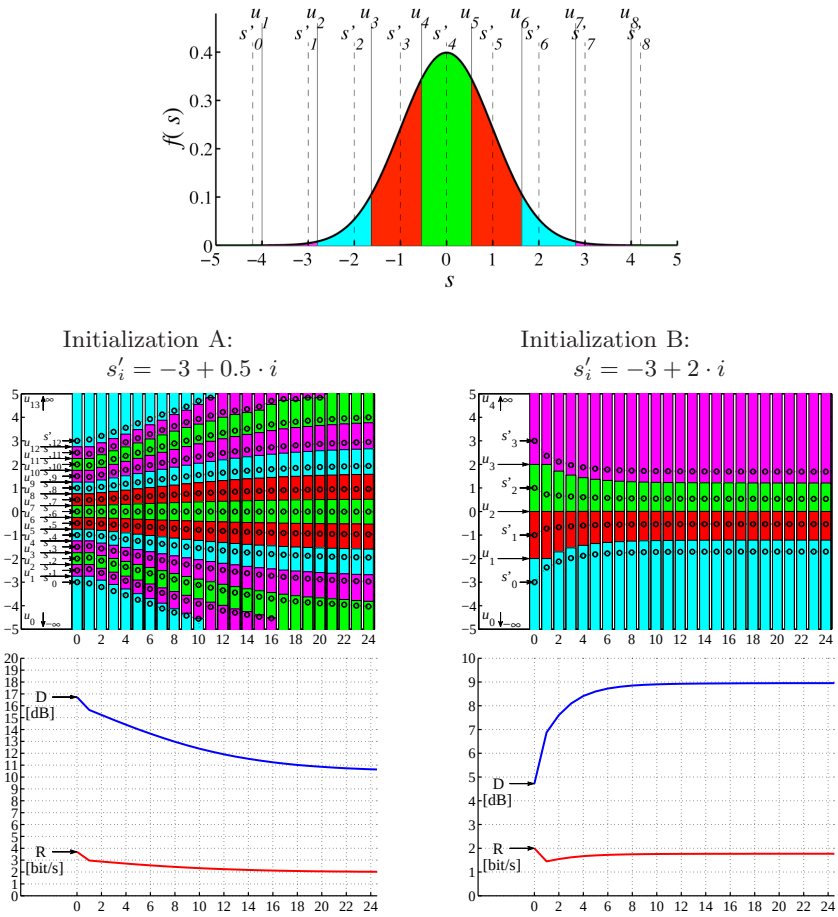


Fig. 5.10 Entropy-constrained Lloyd algorithm for a Gaussian pdf with unit variance and two initializations: (top) final reconstruction levels and decision thresholds; (middle) reconstruction levels and decision thresholds as function of the iteration step; (bottom) overall distortion D and rate R, measured as entropy, as function of the iteration step.

5.2.3 High-Rate Operational Distortion Rate Functions

In general, it is impossible to analytically state the operational distortion rate function for optimized quantizer designs. One of the few exceptions is the uniform distribution, for which the operational distortion rate function for all discussed quantizer designs is given in (5.17). For stationary input processes with continuous random variables, we can, however, derive the asymptotic operational distortion rate func-

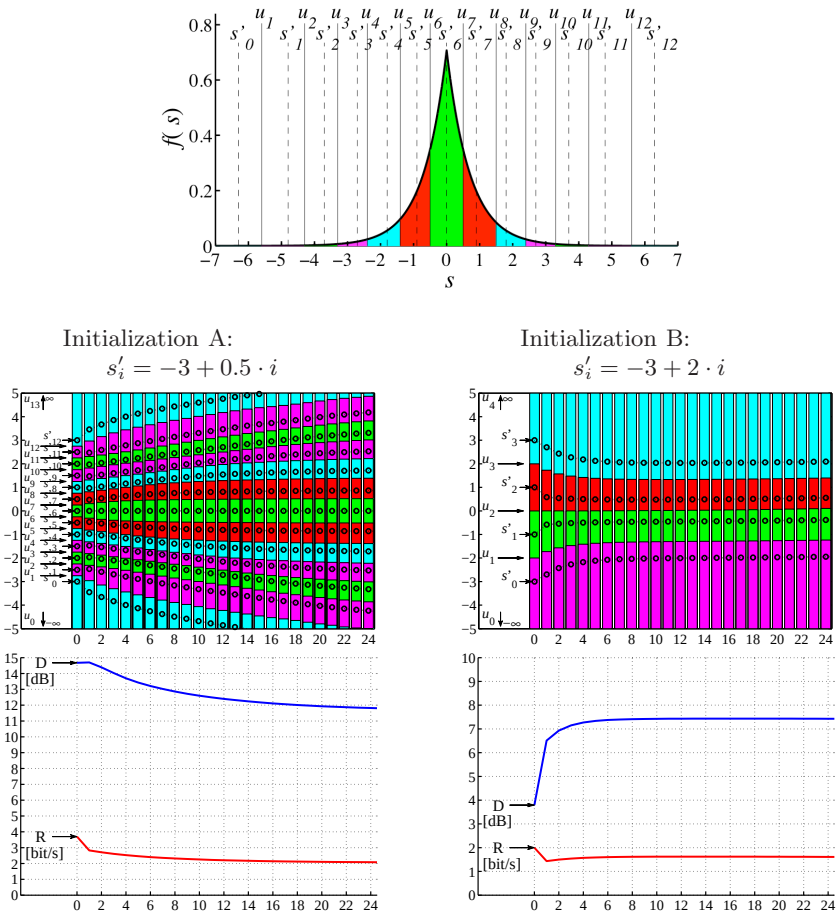


Fig. 5.11 Entropy-constrained Lloyd algorithm for a Laplacian pdf with unit variance and two initializations: (top) final reconstruction levels and decision thresholds; (middle) reconstruction levels and decision thresholds as function of the iteration step; (bottom) overall distortion D and rate R, measured as entropy, as function of the iteration step.

tions for very high rates ($R \to \infty$) or equivalently for small distortions ($D \to 0$). The resulting relationships are referred to as *high-rate approximations* and approach the true operational distortion rate functions as the rate approaches infinity. We remember that as the rate approaches infinity, the (information) distortion rate function approaches the Shannon lower bound. Hence, for high rates, the performance of a quantizer design can be evaluated by comparing the high rate approximation of the operational distortion rate function with the Shannon lower bound.

The general assumption that we use for deriving high-rate approximations is that the sizes Δ_i of the quantization intervals $[u_i, u_{i+1})$ are so small that the marginal pdf $f(s)$ of a continuous input process is nearly constant inside each interval,

$$f(s) \approx f(s'_i) \qquad \text{for} \qquad s \in [u_i, u_{i+1}). \tag{5.36}$$

The probabilities of the reconstruction levels can be approximated by

$$p(s'_i) = \int_{u_i}^{u_{i+1}} f(s)\, ds \approx (u_{i+1} - u_i) f(s'_i) = \Delta_i \cdot f(s'_i). \tag{5.37}$$

For the average distortion D, we obtain

$$D = E\{d(S, Q(S))\} \approx \sum_{i=0}^{K-1} f(s'_i) \int_{u_i}^{u_{i+1}} (s - s'_i)^2\, ds. \tag{5.38}$$

An integration of the right side of (5.38) yields

$$D \approx \frac{1}{3} \sum_{i=0}^{K-1} f(s'_i) \left((u_{i+1} - s'_i)^3 - (u_i - s'_i)^3 \right). \tag{5.39}$$

For each quantization interval, the distortion is minimized if the term $(u_{i+1} - s'_i)^3$ is equal to the term $(u_i - s'_i)^3$, which yields

$$s'_i = \frac{1}{2}(u_i + u_{i+1}). \tag{5.40}$$

By inserting (5.40) into (5.39), we obtain the following expression for the average distortion at high rates,

$$D \approx \frac{1}{12} \sum_{i=0}^{K-1} f(s'_i)\, \Delta_i^3 = \frac{1}{12} \sum_{i=0}^{K-1} p(s'_i)\, \Delta_i^2. \tag{5.41}$$

For deriving the asymptotic operational distortion rate functions, we will use the expression (5.41) with equality, but keep in mind that it is only asymptotically correct for $\Delta_i \to 0$.

PCM Quantization. For PCM quantization of random processes with a finite amplitude range of width A, we can directly insert the expression (5.14) into the distortion approximation (5.41). Since $\sum_{i=0}^{K-1} p(s'_i)$ is equal to 1, this yields the asymptotic operational distortion rate function

$$D_{\text{PCM}}(R) = \frac{1}{12} A^2\, 2^{-2R}. \tag{5.42}$$

Scalar Quantizers with Fixed-Length Codes. In order to derive the asymptotic operational distortion rate function for optimal scalar quantizers in combination with fixed-length codes, we again start with the distortion approximation in (5.41). By using the relationship $\sum_{i=0}^{K-1} K^{-1} = 1$, it can be reformulated as

$$D = \frac{1}{12} \sum_{i=0}^{K-1} f(s'_i)\Delta_i^3 = \frac{1}{12} \left(\left(\sum_{i=0}^{K-1} f(s'_i)\Delta_i^3 \right)^{\frac{1}{3}} \cdot \left(\sum_{i=0}^{K-1} \frac{1}{K} \right)^{\frac{2}{3}} \right)^3 . \quad (5.43)$$

Using Hölders inequality

$$\alpha + \beta = 1 \quad \Rightarrow \quad \left(\sum_{i=a}^{b} x_i \right)^{\alpha} \cdot \left(\sum_{i=a}^{b} y_i \right)^{\beta} \geq \sum_{i=a}^{b} x_i^{\alpha} \cdot y_i^{\beta} \quad (5.44)$$

with equality if and only if x_i is proportional to y_i, it follows

$$D \geq \frac{1}{12} \left(\sum_{i=0}^{K-1} f(s'_i)^{\frac{1}{3}} \cdot \Delta_i \cdot \left(\frac{1}{K} \right)^{\frac{2}{3}} \right)^3 = \frac{1}{12\,K^2} \left(\sum_{i=0}^{K-1} \sqrt[3]{f(s'_i)}\,\Delta_i \right)^3 . \quad (5.45)$$

Equality is achieved if the terms $f(s'_i)\,\Delta_i^3$ are proportional to $1/K$. Hence, the average distortion for high rates is minimized if all quantization intervals have the same contribution to the overall distortion D.

We have intentionally chosen $\alpha = 1/3$, in order to obtain an expression of the sum in which Δ_i has no exponent. Remembering that the used distortion approximation is asymptotically valid for small intervals Δ_i, the summation in (5.45) can be written as integral,

$$D = \frac{1}{12K^2} \left(\int_{-\infty}^{\infty} \sqrt[3]{f(s)}\,\mathrm{d}s \right)^3 . \quad (5.46)$$

As discussed in sec. 5.2.1, the rate R for a scalar quantizer with fixed-length codes is given by $R = \log_2 K$. This yields the following asymptotic operational distortion rate function for optimal scalar quantizers with fixed-length codes,

$$D_F(R) = \sigma^2 \cdot \varepsilon_F^2 \cdot 2^{-2R} \quad \text{with} \quad \varepsilon_F^2 = \frac{1}{\sigma^2} \left(\int_{-\infty}^{\infty} \sqrt[3]{f(s)}\,\mathrm{d}s \right)^3 , \quad (5.47)$$

where the factor ε_F^2 only depends on the marginal pdf $f(s)$ of the input process. The result (5.47) was reported by Panter and Dite in [59] and is also referred to as the *Panter and Dite formula*.

Scalar Quantizers with Variable-Length Codes. In sec. 5.2.2, we have discussed that the rate R for an optimized scalar quantizer with variable-length codes can be approximated by the entropy $H(S')$ of the output random variables S'. We ignore that, for the quantization of sources with memory, the output samples are not mutually independent and hence a lossless code that employs the dependencies between the output samples may achieve a rate below the scalar entropy $H(S')$.

By using the entropy $H(S')$ of the output random variables S' as approximation for the rate R and applying the high-rate approximation $p(s'_i) = f(s'_i)\,\Delta_i$, we obtain

$$\begin{aligned} R \;=\; H(S') &= -\sum_{i=0}^{K-1} p(s'_i)\log_2 p(s'_i) = -\sum_{i=0}^{K-1} f(s'_i)\Delta_i \log_2(f(s'_i)\Delta_i) \\ &= -\sum_{i=0}^{K-1} f(s'_i)\log_2 f(s'_i)\Delta_i - \sum_{i=0}^{K-1} f(s'_i)\Delta_i \log_2 \Delta_i. \end{aligned} \tag{5.48}$$

Since we investigate the asymptotic behavior for small interval sizes Δ_i, the first term in (5.48) can be formulated as an integral, which actually represents the differential entropy $h(S)$, yielding

$$\begin{aligned} R &= -\int_{-\infty}^{\infty} f(s)\,\log_2 f(s)\,\mathrm{d}s - \sum_{i=0}^{K-1} p(s'_i)\,\log_2 \Delta_i \\ &= h(S) - \frac{1}{2}\sum_{i=0}^{K-1} p(s'_i)\,\log_2 \Delta_i^2. \end{aligned} \tag{5.49}$$

We continue with applying Jensen's inequality for convex functions $\varphi(x)$, such as $\varphi(x) = -\log_2 x$, and positive weights a_i,

$$\varphi\left(\sum_{i=0}^{K-1} a_i\,x_i\right) \le \sum_{i=0}^{K-1} a_i\,\varphi(x_i) \qquad \text{for} \qquad \sum_{i=0}^{K-1} a_i = 1. \tag{5.50}$$

By additionally using the distortion approximation (5.41), we obtain

$$R \;\ge\; h(S) - \frac{1}{2}\log_2\left(\sum_{i=0}^{K-1} p(s'_i)\,\Delta_i^2\right) = h(S) - \frac{1}{2}\log_2(12\,D). \tag{5.51}$$

In Jensen's inequality (5.50), equality is obtained if and only if all x_i have the same value. Hence, in the high-rate case, the rate R for a given

distortion is minimized if the quantization step sizes Δ_i are constant. In this case, the quantization is also referred to as *uniform quantization*. The asymptotic operational distortion rate function for optimal scalar quantizers with variable-length codes is given by

$$D_V(R) = \sigma^2 \cdot \varepsilon_V^2 \cdot 2^{-2R} \quad \text{with} \quad \varepsilon_V^2 = \frac{2^{2\,h(S)}}{12\,\sigma^2}. \tag{5.52}$$

Similarly as for the Panter and Dite formula, the factor ε_F^2 only depends on the marginal pdf $f(s)$ of the input process. This result (5.52) was established by Gish and Pierce in [17] using variational calculus and is also referred to as *Gish and Pierce formula*. The use of Jensen's inequality to obtain the same result was first published in [27].

Comparison of the Asymptotic Distortion Rate Functions. We now compare the asymptotic operational distortion rate functions for the discussed quantizer designs with the Shannon lower bound (SLB) for iid sources. All high-rate approximations and also the Shannon lower bound can be written as

$$D_X(R) = \varepsilon_X^2 \cdot \sigma^2 \cdot 2^{-2R}, \tag{5.53}$$

where the subscript X stands for optimal scalar quantizers with fixed-length codes (F), optimal scalar quantizers with variable-length codes (V), or the Shannon lower bound (L). The factors ε_X^2 depend only on the pdf $f(s)$ of the source random variables. For the high-rate approximations, ε_F^2 and ε_V^2 are given by (5.47) and (5.52), respectively. For the Shannon lower bound, ε_L^2 is equal to $2^{2\,h(S)}/(2\pi e)$ as can be easily derived from (4.68). Table 5.1 provides an overview of the various factors ε_X^2 for three example distributions.

If we reformulate (5.53) as signal-to-noise ratio (SNR), we obtain

$$\mathrm{SNR}_X(R) = 10\,\log_{10}\frac{\sigma^2}{D_X(R)} = -10\,\log_{10}\varepsilon_X^2 + R \cdot 20\,\log_{10} 2. \tag{5.54}$$

For all high-rate approximations including the Shannon lower bound, the SNR is a linear function of the rate with a slope of $20\log_{10} 2 \approx 6$. Hence, for high rates the MSE distortion decreases by approximately 6 dB per bit, independently of the source distribution.

	Shannon Lower Bound (SLB)	Panter & Dite (Pdf-Opt w. FLC)	Gish & Pierce (Uniform Q. w. VLC)
Uniform pdf	$\frac{6}{\pi e} \approx 0.7$	1 (1.53 dB to SLB)	1 (1.53 dB to SLB)
Laplacian pdf	$\frac{e}{\pi} \approx 0.86$	$\frac{9}{2} = 4.5$ (7.1 dB to SLB)	$\frac{e^2}{6} \approx 1.23$ (1.53 dB to SLB)
Gaussian pdf	1	$\frac{\sqrt{3}\pi}{2} \approx 2.72$ (4.34 dB to SLB)	$\frac{\pi e}{6} \approx 1.42$ (1.53 dB to SLB)

Table 5.1 Comparison of Shannon lower bound and the high-rate approximations for optimal scalar quantization with fixed-length as well as with variable-length codes.

A further remarkable fact is obtained by comparing the asymptotic operational distortion rate function for optimal scalar quantizers for variable-length codes with the Shannon lower bound. The ratio $D_V(R)/D_L(R)$ is constant and equal to $\pi e/6 \approx 1.53$ dB. The corresponding rate difference $R_V(D) - R_L(D)$ is equal to $\frac{1}{2}\log_2(\pi e/6) \approx 0.25$. At high rates, the distortion of an optimal scalar quantizer with variable-length codes is only 1.53 dB larger than the Shannon lower bound. And for low distortions, the rate increase with respect to the Shannon lower bounds is only 0.25 bit per sample. Due to this fact, scalar quantization with variable-length coding is extensively used in modern video coding.

5.2.4 Approximation for Distortion Rate Functions

The asymptotic operational distortion rate functions for scalar quantizers that we have derived in sec. 5.2.3 can only be used as approximations for high rates. For several optimization problems, it is however desirable to have a simple and reasonably accurate approximation of the distortion rate function for the entire range of rates. In the following, we attempt to derive such an approximation for the important case of entropy-constrained scalar quantization (ECSQ).

If we assume that the optimal entropy-constrained scalar quantizer for a particular normalized distribution (zero mean and unit variance)

and its operational distortion rate function $g(R)$ are known, the optimal quantizer for the same distribution but with different mean and variance can be constructed by an appropriate shifting and scaling of the quantization intervals and reconstruction levels. The distortion rate function $D(R)$ of the resulting scalar quantizer can then be written as

$$D(R) = \sigma^2 \cdot g(R), \tag{5.55}$$

where σ^2 denotes the variance of the input distribution. Hence, it is sufficient to derive an approximation for the normalized operational distortion rate function $g(R)$.

For optimal ECSQ, the function $g(R)$ and its derivative $g'(R)$ should have the following properties:

- If no information is transmitted, the distortion should be equal to the variance of the input signal,

$$g(0) = 1. \tag{5.56}$$

- For high rates, $g(R)$ should be asymptotically tight to the high-rate approximation,

$$\lim_{R\to\infty} \frac{\varepsilon_V^2 \cdot 2^{-2R}}{g(R)} = 1. \tag{5.57}$$

- For ensuring the mathematical tractability of optimization problems the derivative $g'(R)$ should be continuous.
- An increase in rate should result in a distortion reduction,

$$g'(R) < 0 \qquad \text{for} \qquad R \in [0, \infty). \tag{5.58}$$

A function that satisfies the above conditions is

$$g(R) = \frac{\varepsilon_V^2}{a} \cdot \ln(a \cdot 2^{-2R} + 1). \tag{5.59}$$

The factor a is chosen in a way that $g(0)$ is equal to 1. By numerical optimization, we obtained $a = 0.9519$ for the Gaussian pdf and $a = 0.5$ for the Laplacian pdf. For proving that condition (5.57) is fulfilled, we can substitute $x = 2^{-2R}$ and develop the Taylor series of the resulting function

$$g(x) = \frac{\varepsilon^2}{a} \ln(a \cdot x + 1) \tag{5.60}$$

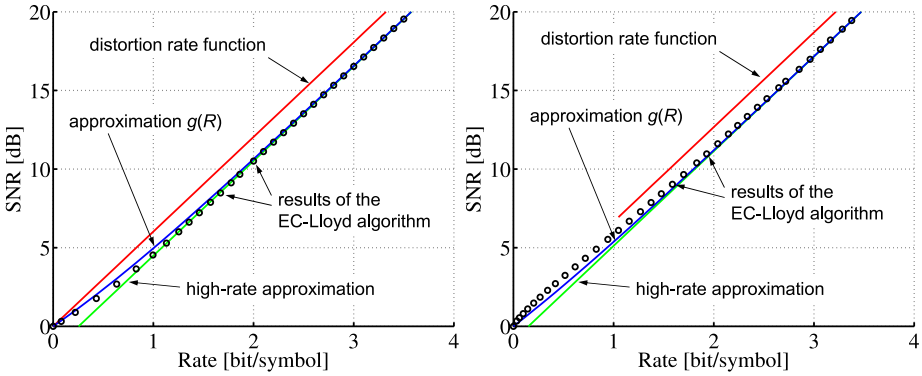


Fig. 5.12 Operational distortion rate functions for a Gaussian (left) and Laplacian (right) pdf with unit variance. The diagrams show the (information) distortion rate function, the high-rate approximation $\varepsilon_V^2\, 2^{-2R}$, and the approximation $g(R)$ given in (5.59). Additionally, results of the EC-Lloyd algorithm with the rate being measured as entropy are shown.

around $x_0 = 0$, which gives

$$g(x) \approx g(0) + g'(0) \cdot x = \varepsilon_V^2 \cdot x. \tag{5.61}$$

Since the remaining terms of the Taylor series are negligible for small values of x (large rates R), (5.59) approaches the high-rate approximation $\varepsilon_V^2\, 2^{-2R}$ as the rate R approaches infinity. The first derivative of (5.59) is given by

$$g'(R) = -\frac{\varepsilon^2 \cdot 2 \ln 2}{a + 2^{2R}}. \tag{5.62}$$

It is a continuous and always less than zero.

The quality of the approximations for the operational distortion rate functions of an entropy-constrained quantizer for a Gaussian and Laplacian pdf is illustrated in Fig. 5.12. For the Gaussian pdf, the approximation (5.59) provides a sufficiently accurate match to the results of the entropy-constrained Lloyd algorithm and will be used later. For the Laplacian pdf, the approximation is less accurate for low bit rates.

5.2.5 Performance Comparison for Gaussian Sources

In the following, we compare the rate distortion performance of the discussed scalar quantizers designs with the rate distortion bound for unit-variance stationary Gauss-Markov sources with $\rho = 0$ and $\rho = 0.9$.

The distortion rate functions for both sources, the operational distortion rates function for PCM (uniform, fixed-rate), the Lloyd design, and the entropy-constraint Lloyd design (EC-Lloyd), as well as the Panter & Dite and Gish & Pierce asymptotes are depicted in Fig. 5.13. The rate for quantizers with fixed-length codes is given by the binary logarithm of the quantizer size K. For quantizers with variable-length codes, it is measured as the entropy of the reconstruction levels.

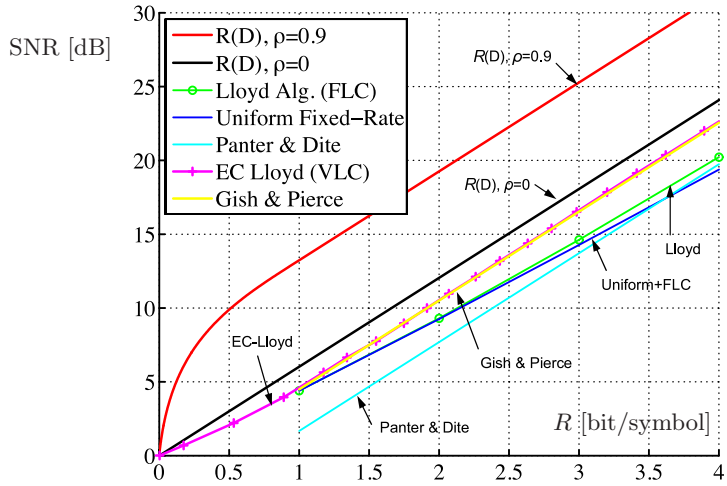


Fig. 5.13 Comparison of the rate distortion performance for Gaussian sources.

The scalar quantizer designs behave identical for both sources as only the marginal pdf $f(s)$ is relevant for the quantizer design algorithms. For high rates, the entropy-constrained Lloyd design and the Gish & Pierce approximation yield an SNR that is 1.53 dB smaller than the (information) distortion rate function for the Gauss-Markov source with $\rho = 0$. The rate distortion performance of the quantizers with fixed-length codes is worse, particularly for rates above 1 bit per sample. It is, however, important to note that it cannot be concluded that the Lloyd algorithm yields a worse performance than the entropy-constrained Lloyd algorithm. Both quantizers are (locally) optimal with respect to their application area. The Lloyd algorithm results an optimized quantizer for fixed-length coding, while the entropy-constrained Lloyd algorithm yields on optimized quantizer for variable-length coding (with an average codeword length close to the entropy).

The distortion rate function for the Gauss-Markov source with $\rho = 0.9$ is far away from the operational distortion rate functions of the investigated scalar quantizer designs. The reason is that we assumed a lossless coding γ that achieves a rate close to the entropy $H(S')$ of the output process. A combination of scalar quantization and advanced lossless coding techniques that exploit dependencies between the output samples is discussed in the next section.

5.2.6 Scalar Quantization for Sources with Memory

In the previous sections, we concentrated on combinations of scalar quantization with lossless coding techniques that do not exploit dependencies between the output samples. As a consequence, the rate distortion performance did only depend on the marginal pdf of the input process, and for stationary sources with memory the performance was identical to the performance for iid sources with the same marginal distribution. If we, however, apply scalar quantization to sources with memory, the output samples are not independent. The dependencies can be exploited by advanced lossless coding techniques such as conditional Huffman codes, block Huffman codes, or arithmetic codes that use conditional pmfs in the probability modeling stage.

The design goal of Lloyd quantizers was to minimize the distortion for a quantizer of a given size K. Hence, the Lloyd quantizer design does not change for source with memory. But the design of the entropy-constrained Lloyd quantizer can be extended by considering advanced entropy coding techniques. The conditions for the determination of the reconstruction levels and interval boundaries (given the decision thresholds and average codeword lengths) do not change, only the determination of the average codeword lengths in step 4 of the entropy-constrained Lloyd algorithm needs to be modified. We can design a lossless code such as a conditional or block Huffman code based on the joint pmf of the output samples (which is given by the joint pdf of the input source and the decision thresholds) and determine the resulting average codeword lengths. But, following the same arguments as in sec. 5.2.2, we can also approximate the average codeword lengths based on the corresponding conditional entropy or block entropy.

For the following consideration, we assume that the input source is stationary and that its joint pdf for N successive samples is given by $f_N(\boldsymbol{s})$. If we employ a conditional lossless code (conditional Huffman code or arithmetic code) that exploits the conditional pmf of a current output sample S' given the last N output samples $\boldsymbol{S'}$, the average codeword lengths $\bar{\ell}(s'_i)$ can be set equal to the ratio of the conditional entropy $H(S'|\boldsymbol{S'})$ and the symbol probability $p(s'_i)$,

$$\bar{\ell}(s'_i) = \frac{H(S'|\boldsymbol{S'})}{p(s'_i)} = -\frac{1}{p(s'_i)} \sum_{k=0}^{K^N-1} p_{N+1}(s'_i, \boldsymbol{s'}_k) \log_2 \frac{p_{N+1}(s'_i, \boldsymbol{s'}_k)}{p_N(\boldsymbol{s'}_k)}, \quad (5.63)$$

where k is an index that indicates any of the K^N combinations of the last N output samples, p is the marginal pmf of the output samples, and p_N and p_{N+1} are the joint pmfs for N and $N+1$ successive output samples, respectively. It should be noted that the argument of the logarithm represents the conditional pmf for an output sample S' given the N preceding output samples $\boldsymbol{S'}$.

Each joint pmf for N successive output samples, including the marginal pmf p with $N=1$, is determined by the joint pdf f_N of the input source and the decision thresholds,

$$p_N(\boldsymbol{s'}_k) = \int_{\boldsymbol{u}_k}^{\boldsymbol{u}_{k+1}} f_N(\boldsymbol{s}) \, \mathrm{d}\boldsymbol{s}, \quad (5.64)$$

where $\boldsymbol{u}_k$ and $\boldsymbol{u}_{k+1}$ represent the ordered sets of lower and upper interval boundaries for the vector $\boldsymbol{s'}_k$ of output samples. Hence, the average codeword length $\bar{\ell}(s'_i)$ can be directly derived based on the joint pdf for the input process and the decision thresholds. In a similar way, the average codeword lengths for block codes of N samples can be approximated based on the block entropy for N successive output samples.

We now investigate the asymptotic operational distortion rate function for high rates. If we again assume that we employ a conditional lossless code that exploits the conditional pmf using the preceding N output samples, the rate R can be approximated by the corresponding conditional entropy $H(S_n|S_{n-1}, \cdots, S_{n-N})$,

$$R = -\sum_{i=0}^{K-1} \sum_{k=0}^{K^N-1} p_{N+1}(s'_i, \boldsymbol{s'}_k) \log_2 \frac{p_{N+1}(s'_i, \boldsymbol{s'}_k)}{p_N(\boldsymbol{s'}_k)}. \quad (5.65)$$

For small quantization intervals Δ_i (high rates), we can assume that the joint pdfs f_N for the input sources are nearly constant inside each N-dimensional hypercube given by a combination of quantization intervals, which yields the approximations

$$p_N(\boldsymbol{s'}_k) = f_N(\boldsymbol{s'}_k)\,\boldsymbol{\Delta}_k \quad \text{and} \quad p_{N+1}(s'_i, \boldsymbol{s'}_k) = f_{N+1}(s'_i, \boldsymbol{s'}_k)\,\boldsymbol{\Delta}_k\,\Delta_i, \tag{5.66}$$

where $\boldsymbol{\Delta}_k$ represents the Cartesian product of quantization interval sizes that are associated with the vector of reconstruction levels $\boldsymbol{s'}_k$. By inserting these approximations in (5.65), we obtain

$$\begin{aligned} R \;=\; & -\sum_{i=0}^{K-1}\sum_{k=0}^{K^N-1} f_{N+1}(s'_i, \boldsymbol{s'}_k)\,\boldsymbol{\Delta}_k\,\Delta_i\,\log_2 \frac{f_{N+1}(s'_i, \boldsymbol{s'}_k)}{f_N(\boldsymbol{s'}_k)} \\ & -\sum_{i=0}^{K-1}\sum_{k=0}^{K^N-1} f_{N+1}(s'_i, \boldsymbol{s'}_k)\,\boldsymbol{\Delta}_k\,\Delta_i\,\log_2 \Delta_i. \end{aligned} \tag{5.67}$$

Since we consider the asymptotic behavior for infinitesimal quantization intervals, the sums can be replaced by integrals, which gives

$$\begin{aligned} R \;=\; & -\int_{\mathcal{R}}\int_{\mathcal{R}^N} f_{n+1}(s, \boldsymbol{s})\,\log_2 \frac{f_{n+1}(s, \boldsymbol{s})}{f_N(\boldsymbol{s})}\,\mathrm{d}\boldsymbol{s}\,\mathrm{d}s \\ & -\sum_{i=0}^{K-1}\left(\int_{\mathcal{R}^N} f_{n+1}(s'_i, \boldsymbol{s})\,\mathrm{d}\boldsymbol{s}\right)\Delta_i\,\log_2 \Delta_i. \end{aligned} \tag{5.68}$$

The first integral (including the minus sign) is the conditional differential entropy $h(S_n|S_{n-1}, \cdots, S_{n-N})$ for an input sample given the preceding N input symbols and the second integral is the value $f(s'_i)$ of marginal pdf of the input source. Using the high rate approximation $p(s'_i) = f(s'_i)\Delta_i$, we obtain

$$R = h(S_n|S_{n-1}, \cdots, S_{n-N}) - \frac{1}{2}\sum_{i=0}^{K-1} p(s'_i)\,\log_2 \Delta_i^2, \tag{5.69}$$

which is similar to (5.49). In the same way as for (5.49) in sec. 5.2.3, we can now apply Jensen's inequality and then insert the high rate approximation (5.41) for the MSE distortion measure. As a consequence

of Jensen's inequality, we note that also for conditional lossless codes, the optimal quantizer design for high rates has uniform quantization steps sizes. The asymptotic operational distortion rate function for an optimum quantizer with conditional lossless codes is given by

$$D_C(R) = \frac{1}{12} \cdot 2^{h(S_n|S_{n-1},\cdots,S_{n-N})} \cdot 2^{-2R}. \tag{5.70}$$

In comparison to the Gish & Pierce asymptote (5.52), the first-order differential entropy $h(S)$ is replaced by the conditional entropy given the N preceding input samples.

In a similar way, we can also derive the asymptotic distortion rate function for block entropy codes (as the block Huffman code) of size N. We obtain the result that also for block entropy codes, the optimal quantizer design for high rates has uniform quantization step sizes. The corresponding asymptotic operational distortion rate function is

$$D_B(R) = \frac{1}{12} \cdot 2^{\frac{h(S_n,\cdots,S_{n+N-1})}{N}} \cdot 2^{-2R}, \tag{5.71}$$

where $h(S_n, \ldots, S_{n+N-1})$ denotes the joint differential entropy for N successive input symbols.

The achievable distortion rate function depends on the complexity of the applied lossless coding technique (which is basically given by the parameter N). For investigating the asymptotically achievable operational distortion rate function for arbitrarily complex entropy coding techniques, we take the limit for $N \to \infty$, which yields

$$D^{\infty}(R) = \frac{1}{12} \cdot 2^{\bar{h}(\boldsymbol{S})} \cdot 2^{-2R}, \tag{5.72}$$

where $\bar{h}(\boldsymbol{S})$ denotes the differential entropy rate of the input source. A comparison with the Shannon lower bound (4.65) shows that the asymptotically achievable distortion for high rates and arbitrarily complex entropy coding is 1.53 dB larger than the fundamental performance bound. The corresponding rate increase is 0.25 bit per sample. It should be noted that this asymptotic bound can only be achieved for high rates. Furthermore, in general, the entropy coding would require the storage of a very large set of codewords or conditional probabilities, which is virtually impossible in real applications.

5.3 Vector Quantization

The investigation of scalar quantization (SQ) showed that it is impossible to achieve the fundamental performance bound using a source coding system consisting of scalar quantization and lossless coding. For high rates the difference to the fundamental performance bound is 1.53 dB or 0.25 bit per sample. This gap can only be reduced if multiple samples are jointly quantized, i.e., by vector quantization (VQ). Although vector quantization is rarely used in video coding, we will give a brief overview in order to illustrate its design, performance, complexity, and the reason for the limitation of scalar quantization.

In N-dimensional vector quantization, an input vector $\boldsymbol{s}$ consisting of N samples is mapped to a set of K reconstruction vectors $\{\boldsymbol{s'}_i\}$. We will generally assume that the input vectors are blocks of N successive samples of a realization of a stationary random process $\{S\}$. Similarly as for scalar quantization, we restrict our considerations to *regular vector quantizers*[5] for which the quantization cells are convex sets[6] and each reconstruction vector is an element of the associated quantization cell. The average distortion and average rate of a vector quantizer are given by (5.5) and (5.7), respectively.

5.3.1 Vector Quantization with Fixed-Length Codes

We first investigate a vector quantizer design that minimizes the distortion D for a given quantizer size K, i.e., the counterpart of the Lloyd quantizer. The necessary conditions for the reconstruction vectors and quantization cells can be derived in the same way as for the Lloyd quantizer in sec. 5.2.1 and are given by

$$\boldsymbol{s'}_i = \arg \min_{\boldsymbol{s'} \in \mathcal{R}^N} E\{d_N(\boldsymbol{S}, \boldsymbol{s'}) \mid \boldsymbol{S} \in \mathcal{C}_i\}\,, \tag{5.73}$$

and

$$Q(\boldsymbol{s}) = \arg \min_{\forall\, \boldsymbol{s'}_i} d_N(\boldsymbol{s}, \boldsymbol{s'}_i). \tag{5.74}$$

[5] Regular quantizers are optimal with respect to the MSE distortion measure.

[6] A set of points in $\mathcal{R}^N$ is convex, if for any two points of the set, all points on the straight line connecting the two points are also elements of the set.

The Linde-Buzo-Gray Algorithm. The extension of the Lloyd algorithm to vector quantization [46] is referred to as *Linde-Buzo-Gray algorithm* (LBG). For a sufficiently large training set $\{\boldsymbol{s}_n\}$ and a given quantizer size K, the algorithm can be stated as follows:

(1) Choose an initial set of reconstruction vectors $\{\boldsymbol{s'}_i\}$.

(2) Associate all samples of the training set $\{\boldsymbol{s}_n\}$ with one of the quantization cells $\mathcal{C}_i$ according to

$$a(\boldsymbol{s}_n) = \arg\min_{\forall i} \; d_N(\boldsymbol{s}_n, \boldsymbol{s'}_i).$$

(3) Update the reconstruction vectors $\{\boldsymbol{s'}_i\}$ according to

$$\boldsymbol{s'}_i = \arg\min_{\boldsymbol{s'} \in \mathcal{R}^N} E\{d_N(\boldsymbol{S}, \boldsymbol{s'}) \mid \alpha(\boldsymbol{S}) = i\}\,,$$

where the expectation value is taken over the training set.

(4) Repeat the previous two steps until convergence.

Examples for the LBG Algorithm. An an example, we designed a 2-d vector quantizer for a Gaussian iid process with unit variance. The selected quantizer size is $K = 16$ corresponding to a rate of 2 bit per (scalar) sample. The chosen initialization as well as the obtained quantization cells and reconstruction vectors after the 8-th and 49-th iteration of the LBG algorithm are illustrated in Fig. 5.14. In Fig. 5.15, the distortion is plotted as function of the iteration step.

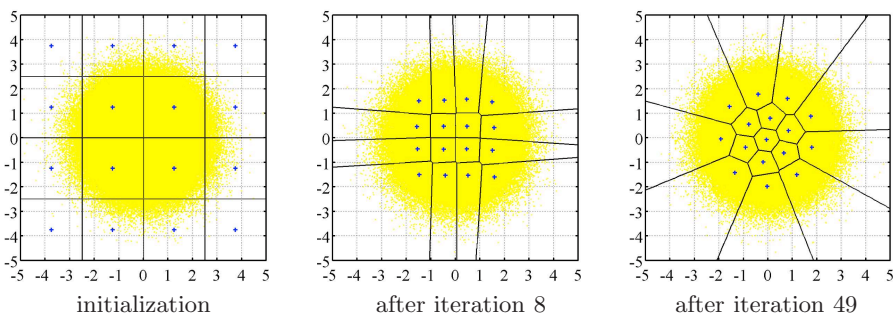


Fig. 5.14 Illustration of the LBG algorithm for a quantizer with $N = 2$ and $K = 16$ and a Gaussian iid process with unit variance. The lines mark the boundaries of the quantization cells, the crosses show the reconstruction vectors, and the light-colored dots represent the samples of the training set.

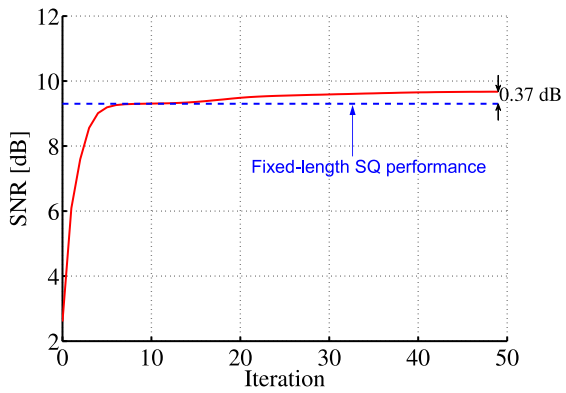


Fig. 5.15 Distortion as function of the iteration step for the LBG algorithm with $N = 2$, $K = 16$, and a Gaussian iid process with unit variance. The dashed line represents the distortion for a Lloyd quantizer with the same rate of $R = 2$ bit per sample.

After the 8-th iteration, the 2-dimensional vector quantizer shows a similar distortion (9.30 dB) as the scalar Lloyd quantizer at the same rate of $R = 2$ bit per (scalar) sample. This can be explained by the fact that the quantization cells are approximately rectangular shaped and that such rectangular cells would also be constructed by a corresponding scalar quantizer (if we illustrate the result for 2 consecutive samples). After the 49-th iteration, the cells of the vector quantizer are shaped in a way that a scalar quantizer cannot create and the SNR is improved to 9.67 dB.

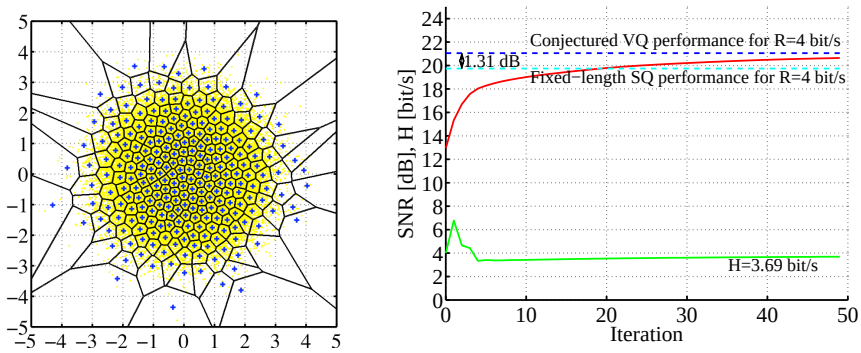


Fig. 5.16 Illustration of the LBG algorithm for a quantizer with $N = 2$ and $K = 256$ and a Gaussian iid process with unit variance: (left) resulting quantization cells and reconstruction vectors after 49 iterations; (right) distortion as function of the iteration step.

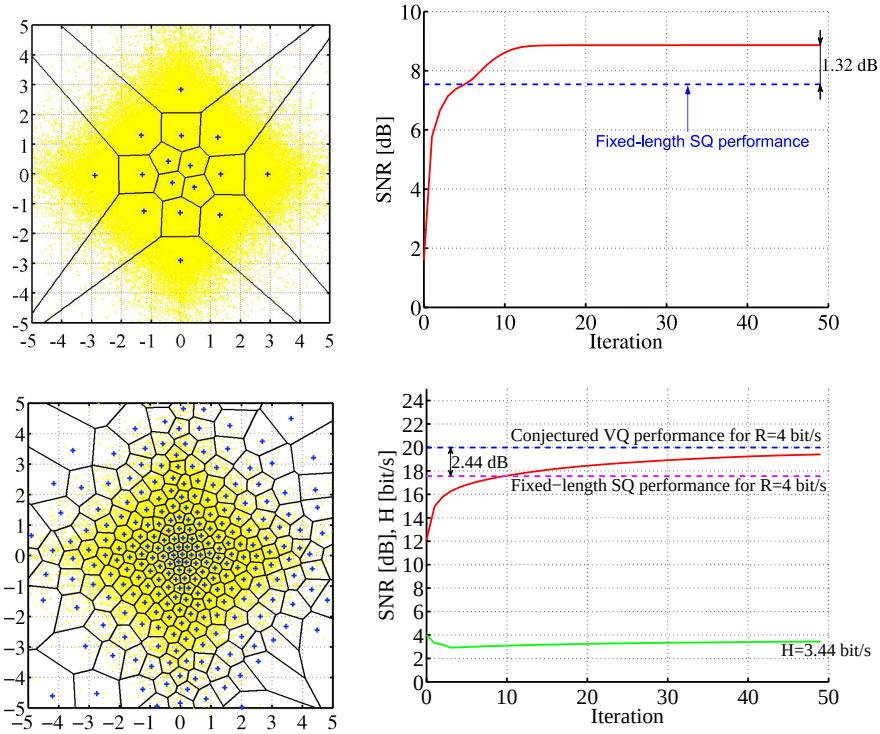


Fig. 5.17 Results of the LBG algorithm for a 2-d VQ with a size of $K = 16$ (top) and $K = 256$ (bottom) for a Laplacian iid source with unit variance.

Fig. 5.16 shows the result of the LBG algorithm for a vector quantizer with $N = 2$ and $K = 256$, corresponding to a rate of $R = 4$ bit per sample, for the Gaussian iid source with unit variance. After the 49-th iteration, the gain for two-dimensional VQ is around 0.9 dB compared to SQ with fixed-length codes resulting in an SNR of 20.64 dB (of conjectured 21.05 dB [50]). The result indicates that at higher bit rates, the gain of VQ relative to SQ with fixed-length codes increases.

Fig. 5.17 illustrates the results for a 2-d VQ design for a Laplacian iid source with unit variance and two different quantizer sizes K. For $K = 16$, which corresponds to a rate of $R = 2$ bit per sample, the SNR is 8.87 dB. Compared to SQ with fixed-length codes at the same rate, a gain of 1.32 dB has been achieved. For a rate of $R = 4$ bit per sample ($K = 256$), the SNR gain is increased to 1.84 dB resulting in an SNR of 19.4 dB (of conjectured 19.99 dB [50]).

5.3.2 Vector Quantization with Variable-Length Codes

For designing a vector quantizer with variable-length codes, we have to minimize the distortion D subject to a rate constraint, which can be effectively done using Lagrangian optimization. Following the arguments in sec. 5.2.2, it is justified to approximate the rate by the entropy $H(Q(\boldsymbol{S}))$ of the output vectors and to set the average codeword lengths equal to $\bar{\ell}(\boldsymbol{s'}_i) = -\log_2 p(\boldsymbol{s'}_i)$. Such a quantizer design is also referred to as *entropy-constrained vector quantizer* (ECVQ). The necessary conditions for the reconstruction vectors and quantization cells can be derived in the same way as for the entropy-constrained scalar quantizer (ECSQ) and are given by (5.73) and

$$Q(\boldsymbol{s}) = \arg\min_{\forall\, \boldsymbol{s'}_i} d_N(\boldsymbol{s}, \boldsymbol{s'}_i) + \lambda\, \bar{\ell}(\boldsymbol{s'}_i). \tag{5.75}$$

The Chou-Lookabaugh-Gray Algorithm. The extension of the entropy-constrained Lloyd algorithm to vector quantization [9] is also referred to as *Chou-Lookabaugh-Gray algorithm* (CLG). For a sufficiently large training set $\{\boldsymbol{s}_n\}$ and a given Lagrange parameter λ, the CLG algorithm can be stated as follows:

(1) Choose an initial quantizer size N and initial sets of reconstruction vectors $\{\boldsymbol{s'}_i\}$ and average codeword lengths $\bar{\ell}(\boldsymbol{s'}_i)$.

(2) Associate all samples of the training set $\{\boldsymbol{s}_n\}$ with one of the quantization cells $\mathcal{C}_i$ according to

$$\alpha(\boldsymbol{s}) = \arg\min_{\forall\, \boldsymbol{s'}_i} d_N(\boldsymbol{s}, \boldsymbol{s'}_i) + \lambda\, \bar{\ell}(\boldsymbol{s'}_i).$$

(3) Update the reconstruction vectors $\{\boldsymbol{s'}_i\}$ according to

$$\boldsymbol{s'}_i = \arg\min_{\boldsymbol{s'} \in \mathcal{R}^N} E\{d_N(\boldsymbol{S}, \boldsymbol{s'}) \mid \alpha(\boldsymbol{S}) = i\}\,,$$

where the expectation value is taken over the training set.

(4) Update the average codeword length $\bar{\ell}(\boldsymbol{s'}_i)$ according to

$$\bar{\ell}(\boldsymbol{s'}_i) = -\log_2 p(\boldsymbol{s'}_i).$$

(5) Repeat the previous three steps until convergence.

Examples for the CLG Algorithm. As examples, we designed a 2-d ECVQ for a Gaussian and Laplacian iid process with unit variance and an average rate, measured as entropy, of $R = 2$ bit per sample. The results of the CLG algorithm are illustrated in Fig. 5.18. The SNR gain compared to an ECSQ design with the same rate is 0.26 dB for the Gaussian and 0.37 dB for the Laplacian distribution.

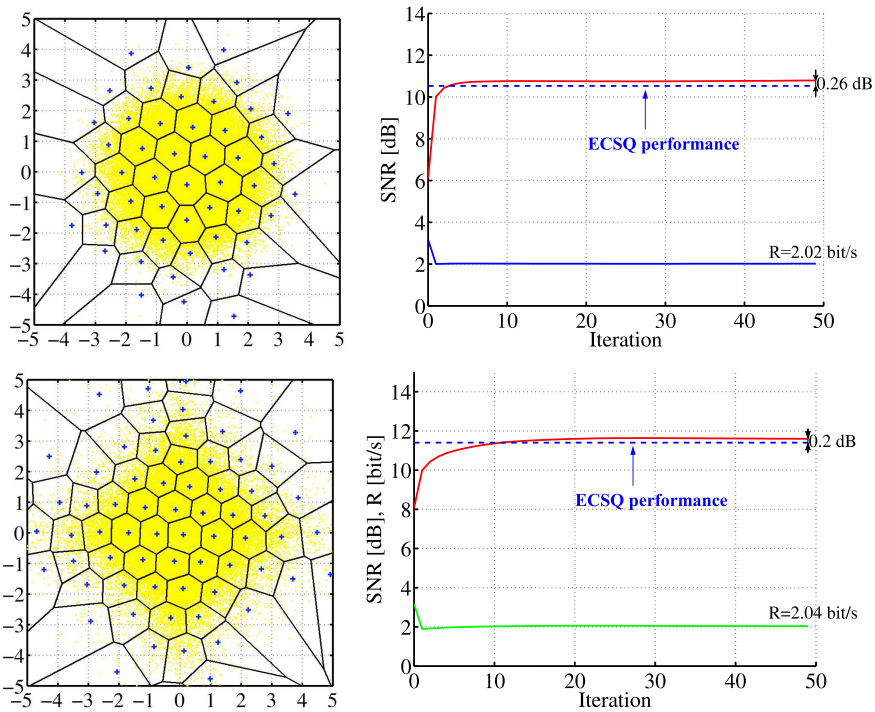


Fig. 5.18 Results of the CLG algorithm for a Gaussian (top) and Laplacian (bottom) iid source with unit variance and a rate (entropy) of $R = 2$ bit per sample. The dashed line in the diagrams on the right shows the distortion for an ECSQ design with the same rate.

5.3.3 The Vector Quantization Advantage

The examples for the LBG and CLG algorithms showed that vector quantization increases the coding efficiency compared to scalar quantization. According to the intuitive analysis in [52], the performance gain can be attributed to three different effects: the space filling advantage, the shape advantage, and the memory advantage. In the following,

we will briefly explain and discuss these advantages. We will see that the space filling advantage is the only effect that can be exclusively achieved with vector quantization. The associated performance gain is bounded to 1.53 dB or 0.25 bit per sample. This bound is asymptotically achieved for large quantizer dimensions and large rates, and corresponds exactly to the gap between the operational rate distortion function for scalar quantization with arbitrarily complex entropy coding and the rate distortion bound at high rates. For a deeper analysis of the vector quantization advantages, the reader is referred to the discussion in [52] and the quantitative analysis in [50].

Space Filling Advantage. When we analyze the results of scalar quantization in higher dimension, we see that the N-dimensional space is partitioned into N-dimensional hyperrectangles (Cartesian products of intervals). This does however not represent the densest packing in $\mathcal{R}^N$. With vector quantization of dimension N, we have extra freedom in choosing the shapes of the quantization cells. The associated increase in coding efficiency is referred to as *space filling advantage*.

The space filling advantage can be observed in the example for the LBG algorithm with $N = 2$ and a Gaussian iid process in Fig. 5.14. After the 8-th iteration, the distortion is approximately equal to the distortion of the scalar Lloyd quantizer with the same rate and the reconstruction cells are approximately rectangular shaped. However, the densest packing in two dimensions is achieved by hexagonal quantization cells. After the 49-th iteration of the LBG algorithm, the quantization cells in the center of the distribution look approximately like hexagons. For higher rates, the convergence toward hexagonal cells is even better visible as can be seen in Figs. 5.16 and 5.17

To further illustrate the space filling advantage, he have conducted another experiment for a uniform iid process with $A = 10$. The operational distortion rate function for scalar quantization is given by $D(R) = \frac{A^2}{12}\, 2^{-2R}$. For a scalar quantizer of size $K = 10$, we obtain a rate (entropy) of 3.32 bit per sample and a distortion of 19.98 dB. The LBG design with $N = 2$ and $K = 100$ is associated with about the same rate. The partitioning converges toward a hexagonal lattice as illustrated in Fig. 5.19 and the SNR is increased to 20.08 dB.

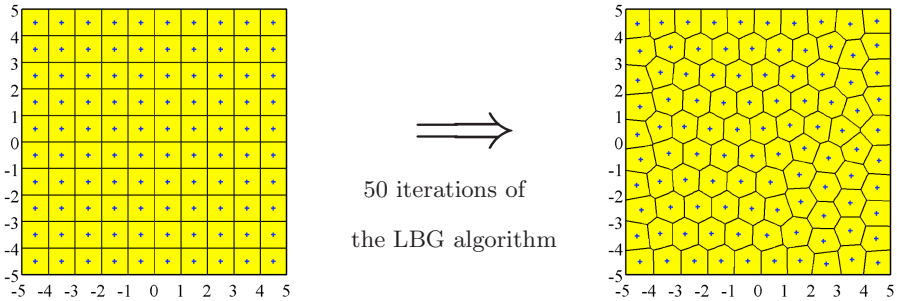


Fig. 5.19 Convergence of LBG algorithm with $N = 2$ toward hexagonal quantization cells for a uniform iid process.

The gain due to choosing the densest packing is independent of the source distribution or any statistical dependencies between the random variables of the input process. The space filling gain is bounded to 1.53 dB, which can be asymptotically achieved for high rates if the dimensionality of the vector quantizer approaches infinity [50].

Shape Advantage. The shape advantage describes the effect that the quantization cells of optimal VQ designs adapt to the shape of the source pdf. In the examples for the CLG algorithm, we have however seen that, even though ECVQ provides a better performance than VQ with fixed-length code, the gain due to VQ is reduced if we employ variable-length coding for both VQ and SQ. When comparing ECVQ

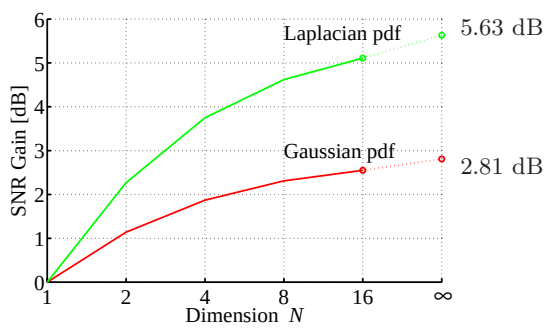


Fig. 5.20 Shape advantage for Gaussian and Laplacian iid sources as function of the vector quantizer dimension N.

with ECSQ for iid sources, the gain of VQ reduces to the space filling advantage, while the shape advantage is exploited by variable-length coding. However, VQ with fixed-length codes can also exploit the gain that ECSQ shows compared to SQ with fixed-length codes [50].

The shape advantage for high rates has been estimated in [50]. Fig. 5.20 shows this gain for Gaussian and Laplacian iid random processes. In practice, the shape advantage is exploited by using scalar quantization in combination with entropy coding techniques such as Huffman coding or arithmetic coding.

Memory Advantage. For sources with memory, there are linear or nonlinear dependencies between the samples. In optimal VQ designs, the partitioning of the N-dimensional space into quantization cells is chosen in a way that these dependencies are exploited. This is illustrated in Fig. 5.21, which shows the ECVQ result of the CLG algorithm for $N = 2$ and a Gauss-Markov process with a correlation factor of $\rho = 0.9$ for two different values of the Lagrange parameter λ.

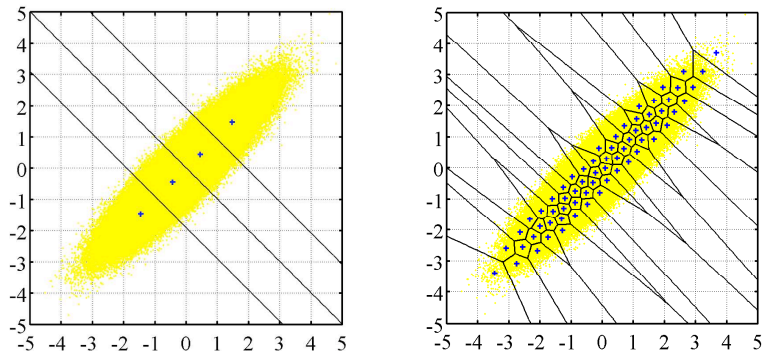


Fig. 5.21 Results of the CLG algorithm with $N = 2$ and two different values of λ for a Gauss-Markov source with $\rho = 0.9$.

An quantitative estimation of the gain resulting from the memory advantage at high rates was done in [50]. Fig. 5.22 shows the memory gain for Gauss-Markov sources with different correlation factors as function of the quantizer dimension N.

For sources with strong dependencies between the samples, such as video signals, the memory gain is much larger than the shape and space

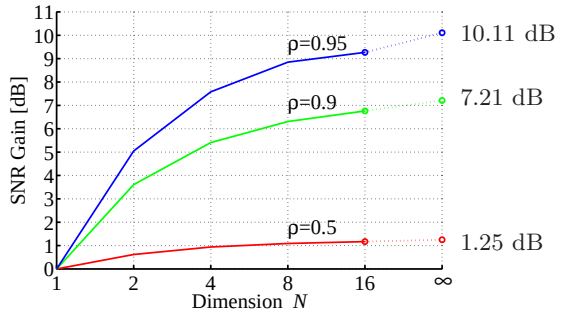


Fig. 5.22 Memory gain as function of the quantizer dimension N for Gauss-Markov sources with different correlation factors ρ.

filling gain. In video coding, a suitable exploitation of the statistical dependencies between samples is one of the most relevant design aspects. The linear dependencies between samples can also be exploited by combining scalar quantization with linear prediction or linear transforms. These techniques are discussed in chapters 6 and 7. By combining scalar quantization with advanced entropy coding techniques, which we discussed in sec. 5.2.6, it is possible to partially exploit both linear as well as nonlinear dependencies.

5.3.4 Performance and Complexity

For further evaluating the performance of vector quantization, we compared the operational rate distortion functions for CLG designs with different quantizer dimensions N to the rate distortion bound and the operational distortion functions for scalar quantizers with fixed-length and variable-length[7] codes. The corresponding rate distortion curves for a Gauss-Markov process with a correlation factor of $\rho = 0.9$ are depicted in Fig. 5.23. For quantizers with fixed-length codes, the rate is given the binary logarithm of the quantizer size K; for quantizers with variable-length codes, the rate is measured as the entropy of the reconstruction levels or reconstruction vectors.

The operational distortion rate curves for vector quantizers of dimensions $N = 2, 5, 10$, and 100, labeled with "VQ, $K = N$(e)", show

[7] In this comparison, it is assumed that the dependencies between the output samples or output vectors are not exploited by the applied lossless coding.

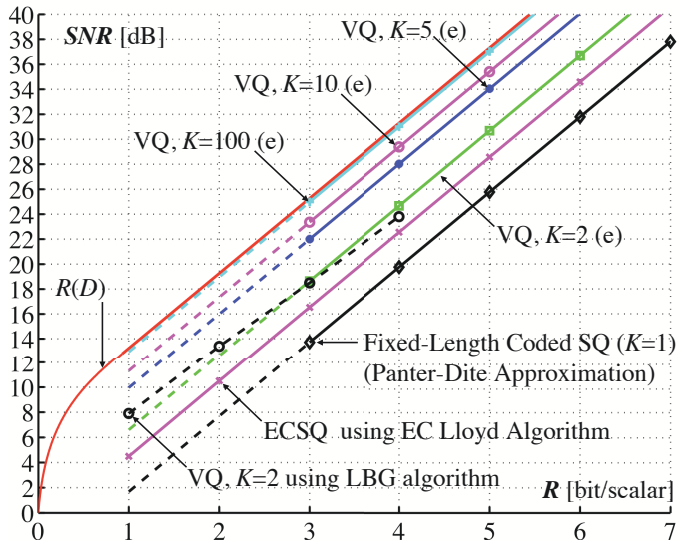


Fig. 5.23 Estimated vector quantization advantage at high rates [50] for a Gauss-Markov source with a correlation factor of $\rho = 0.9$.

the theoretical performance for high rates, which has been estimated in [50]. These theoretical results have been verified for $N = 2$ by designing entropy-constrained vector quantizers using the CLG algorithm. The theoretical vector quantizer performance for a quantizer dimension of $N = 100$ is very close to the distortion rate function of the investigated source. In fact, vector quantization can asymptotically achieve the rate distortion bound as the dimension N approaches infinity. Moreover, vector quantization can be interpreted as the most general lossy source coding system. Each source coding system that maps a vector of N samples to one of K codewords (or codeword sequences) can be designed as vector quantizer of dimension N and size K.

Despite the excellent coding efficiency vector quantization is rarely used in video coding. The main reason is the associated complexity. On one hand, a general vector quantizer requires the storage of a large codebook. This issue becomes even more problematic for systems that must be able to encode and decode sources at different bit rates, as it is required for video codecs. On the other hand, the computationally

complexity for associating an input vector with the best reconstruction vector in rate distortion sense is very large in comparison to the encoding process for scalar quantization that is used in practice. One way to reduce the requirements on storage and computational complexity is to impose structural constraints on the vector quantizer. Examples for such structural constraints include:

- Tree-Structured VQ,
- Transform VQ,
- Multistage VQ,
- Shape-Gain VQ,
- Lattice Codebook VQ,
- Predictive VQ.

In particular, predictive VQ can be seen as a generalization of a number of very popular techniques including motion compensation in video coding. For the actual quantization, video codecs mostly include a simple scalar quantizer with uniformly distributed reconstruction levels (sometimes with a deadzone around zero), which is combined with entropy coding and techniques such as linear prediction or linear transforms in order to exploit the shape of the source distribution and the statistical dependencies of the source. For video coding, the complexity of vector quantizers including those with structural constraints is considered as too large in relation to the achievable performance gains.

5.4 Summary of Quantization

In this chapter, we have discussed quantization starting with scalar quantizers. The Lloyd quantizer that is constructed using an iterative procedure provides the minimum distortion for a given number of reconstruction levels. It is the optimal quantizer design if the reconstruction levels are transmitted using fixed-length codes. The extension of the quantizer design for variable-length codes is achieved by minimizing the distortion D subject to a rate constraint $R < R_{\max}$, which can be formulated as a minimization of a Lagrangian functional $D + \lambda R$. The corresponding iterative design algorithm includes a sufficiently accurate estimation of the codeword lengths that are associated with the

reconstruction levels. Usually the codeword lengths are estimated based on the entropy of the output signal, in which case the quantizer design is also referred to as entropy-constrained Lloyd quantizer.

At high rates, the operational distortion rate functions for scalar quantization with fixed- and variable-length codes as well as the Shannon lower bound can be described by

$$D_X(R) = \sigma^2 \cdot \varepsilon_X^2 \cdot 2^{-2R}, \tag{5.76}$$

where X either indicates the Shannon lower bound or scalar quantization with fixed- or variable-length codes. For a given X, the factors ε_X^2 depend only of the statistical properties of the input source. If the output samples are coded with an arbitrarily complex entropy coding scheme, the difference between the operational distortion rate function for optimal scalar quantization and the Shannon lower bound is 1.53 dB or 0.25 bit per sample at high rates. Another remarkable result is that at high rates, optimal scalar quantization with variable-length codes is achieved if all quantization intervals have the same size.

In the second part of the chapter, we discussed the extension of scalar quantization to vector quantization, by which the rate distortion bound can be asymptotically achieved as the quantizer dimension approaches infinity. The coding efficiency improvements of vector quantization relative to scalar quantization can be attributed to three different effects: the space filling advantage, the shape advantage, and the memory advantage. While the space filling advantage can be only achieved by vector quantizers, the shape and memory advantage can also be exploited by combining scalar quantization with a suitable entropy coding and techniques such as linear prediction and linear transforms.

Despite its superior rate distortion performance, vector quantization is rarely used in video coding applications because of its complexity. Instead, modern video codecs combine scalar quantization with entropy coding, linear prediction, and linear transforms in order to achieve a high coding efficiency at a moderate complexity level.

6

Predictive Coding

In the previous chapter, we investigated the design and rate distortion performance of quantizers. We showed that the fundamental rate distortion bound can be virtually achieved by unconstrained vector quantization of a sufficiently large dimension. However, due to the very large amount of data in video sequences and the real-time requirements that are found in most video coding applications, only low-complex scalar quantizers are typically used in this area. For iid sources, the achievable operational rate distortion function for high rate scalar quantization lies at most 1.53 dB or 0.25 bit per sample above the fundamental rate distortion bound. This represents a suitable trade-off between coding efficiency and complexity. But if there is a large amount of dependencies between the samples of an input signal, as it is the case in video sequences, the rate distortion performance for simple scalar quantizers becomes significantly worse than the rate distortion bound. A source coding system consisting of a scalar quantizer and an entropy coder can exploit the statistical dependencies in the input signal only if the entropy coder uses higher-order conditional or joint probability models. The complexity of such an entropy coder is however close to that of a vector quantizer, so that such a design is unsuitable in practice.

Furthermore, video sequences are highly nonstationary and conditional or joint probabilities for nonstationary sources are typically very difficult to estimate accurately. It is desirable to combine scalar quantization with additional tools that can efficiently exploit the statistical dependencies in a source at a low complexity level. One of such coding concepts is predictive coding, which we will investigate in this chapter. The concepts of prediction and predictive coding are widely used in modern video coding. Well-known examples are intra prediction, motion-compensated prediction, and motion vector prediction.

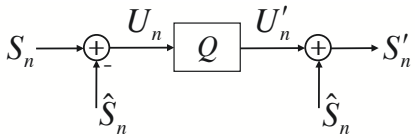


Fig. 6.1 Basic structure of predictive coding.

The basic structure of predictive coding is illustrated in Fig. 6.1 using the notation of random variables. The source samples $\{s_n\}$ are not directly quantized. Instead, each sample s_n is predicted based on previous samples. The prediction value $\hat{s}_n$ is subtracted from the value of the input sample s_n yielding a residual or prediction error sample $u_n = s_n - \hat{s}_n$. The residual sample u_n is then quantized using scalar quantization. The output of the quantizer is a reconstructed value u'_n for the residual sample u_n. At the decoder side, the reconstruction u'_n of the residual sample is added to the predictor $\hat{s}_n$ yielding the reconstructed output sample $s'_n = \hat{s}_n + u'_n$.

Intuitively, we can say that the better the future of a random process is predicted from its past and the more redundancy the random process contains, the less new information is contributed by each successive observation of the process. In the context of predictive coding, the predictors $\hat{s}_n$ should be chosen in a way that they can be easily computed and result in a rate distortion efficiency of the predictive coding system that is as close as possible to the rate distortion bound.

In this chapter, we discuss the design of predictors with the emphasis on linear predictors and analyze predictive coding systems. For further details, the reader is referred to the classic tutorial [51], and the detailed treatments in [75] and [24].

6.1 Prediction

Prediction is a statistical estimation procedure where the value of a particular random variable S_n of a random process $\{S_n\}$ is estimated based on the values of other random variables of the process. Let $\mathcal{B}_n$ be a set of observed random variables. As a typical example, the observation set can represent the N random variables $\mathcal{B}_n = \{S_{n-1}, S_{n-2}, \cdots, S_{n-N}\}$ that precede that random variable S_n to be predicted. The predictor for the random variable S_n is a deterministic function of the observation set $\mathcal{B}_n$ and is denoted by $A_n(\mathcal{B}_n)$. In the following, we will omit this functional notation and consider the prediction of a random variable S_n as another random variable denoted by $\hat{S}_n$,

$$\hat{S}_n = A_n(\mathcal{B}_n). \tag{6.1}$$

The prediction error or residual is given by the difference of the random variable S_n to be predicted and its prediction $\hat{S}_n$. It can also be interpreted as a random variable and is be denoted U_n,

$$U_n = S_n - \hat{S}_n. \tag{6.2}$$

If we predict all random variables of a random process $\{S_n\}$, the sequence of predictions $\{\hat{S}_n\}$ and the sequence of residuals $\{U_n\}$ are random processes. The prediction can then be interpreted as a mapping of an input random process $\{S_n\}$ to an output random process $\{U_n\}$ representing the sequence of residuals as illustrated in Fig. 6.2.

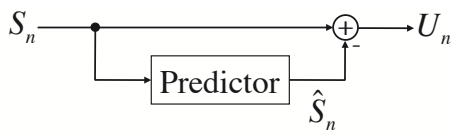


Fig. 6.2 Block diagram of a predictor.

In order to derive optimum predictors, we have to discuss first how the goodness of a predictor can be evaluated. In the context of predictive coding, the ultimate goal is to achieve the minimum distortion between the original and reconstructed samples subject to a given maximum rate. For the MSE distortion measure (or in general for all additive difference distortion measures), the distortion between a vector

of N input samples $\boldsymbol{s}$ and the associated vector of reconstructed samples $\boldsymbol{s'}$ is equal to the distortion between the corresponding vector of residuals $\boldsymbol{u}$ and the associated vector of reconstructed residuals $\boldsymbol{u'}$,

$$d_N(\boldsymbol{s}, \boldsymbol{s'}) = \frac{1}{N}\sum_{i=0}^{N-1}(s_i - s'_i)^2 = \frac{1}{N}\sum_{i=0}^{N-1}(u_i + \hat{s}_i - u'_i - \hat{s}_i)^2 = d_N(\boldsymbol{u}, \boldsymbol{u'}). \tag{6.3}$$

Hence, the operational distortion rate function of a predictive coding systems is equal to the operational distortion rate function for scalar quantization of the prediction residuals. As stated in sec. 5.2.4, the operational distortion rate curve for scalar quantization of the residuals can be stated as $D(R) = \sigma_U^2 \cdot g(R)$, where σ_U^2 is the variance of the residuals and the function $g(R)$ depends only on the type of the distribution of the residuals. Hence, the rate distortion efficiency of a predictive coding system depends on the variance of the residuals and the type of their distribution. We will neglect the dependency on the distribution type and define that a predictor $A_n(\mathcal{B}_n)$ given an observation set $\mathcal{B}_n$ is optimal if it minimizes the variance σ_U^2 of the prediction error. In the literature [51, 75, 24], the most commonly used criterion for the optimality of a predictor is the minimization of the MSE between the input signal and its prediction. This is equivalent to the minimization of the second moment $\epsilon_U^2 = \sigma_U^2 + \mu_U^2$, or the energy, of the prediction error signal. Since the minimization of the second moment ϵ_U^2 implies[1] a minimization of the variance σ_U^2 and the mean μ_U, we will also consider the minimization of the mean squared prediction error ϵ_U^2.

When considering the more general criterion of the mean squared prediction error, the selection of the optimal predictor $A_n(\mathcal{B}_n)$ given an observation set $\mathcal{B}_n$ is equivalent to the minimization of

$$\epsilon_U^2 = E\{U_n^2\} = E\left\{\left(S_n - \hat{S}_n\right)^2\right\} = E\left\{\left(S_n - A_n(\mathcal{B}_n)\right)^2\right\}. \tag{6.4}$$

The solution to this minimization problem is given by the conditional mean of the random variable S_n given the observation set $\mathcal{B}_n$,

$$\hat{S}_n^* = A_n^*(\mathcal{B}_n) = E\{S_n \,|\, \mathcal{B}_n\}. \tag{6.5}$$

[1] We will later prove this statement for linear prediction.

This can be proved by using the formulation

$$\begin{aligned}\epsilon_U^2 &= E\Big\{\big(S_n - E\{S_n \,|\, \mathcal{B}_n\} + E\{S_n \,|\, \mathcal{B}_n\} - A_n(\mathcal{B}_n)\big)^2\Big\} \\ &= E\Big\{\big(S_n - E\{S_n \,|\, \mathcal{B}_n\}\,\big)^2\Big\} + \big(E\{S_n \,|\, \mathcal{B}_n\} - A_n(\mathcal{B}_n)\big)^2 - \\ &\qquad 2\,E\big\{\big(S_n - E\{S_n \,|\, \mathcal{B}_n\}\,\big)\big(E\{S_n \,|\, \mathcal{B}_n\} - A_n(\mathcal{B}_n)\big)\big\}\,. \qquad (6.6)\end{aligned}$$

Since $E\{S_n \,|\, \mathcal{B}_n\}$ and $A_n(\mathcal{B}_n)$ are deterministic functions given the observation set $\mathcal{B}_n$, we can write

$$\begin{aligned}&E\big\{\big(S_n - E\{S_n \,|\, \mathcal{B}_n\}\,\big)\big(E\{S_n \,|\, \mathcal{B}_n\} - A_n(\mathcal{B}_n)\big) \,|\, \mathcal{B}_n\big\} \\ &\qquad = \big(E\{S_n \,|\, \mathcal{B}_n\} - A_n(\mathcal{B}_n)\big) \cdot E\{S_n - E\{S_n \,|\, \mathcal{B}_n\} \,|\, \mathcal{B}_n\} \\ &\qquad = \big(E\{S_n \,|\, \mathcal{B}_n\} - A_n(\mathcal{B}_n)\big) \cdot \big(E\{S_n \,|\, \mathcal{B}_n\} - E\{S_n \,|\, \mathcal{B}_n\}\,\big) \\ &\qquad = 0. \qquad (6.7)\end{aligned}$$

By using the iterative expectation rule $E\{E\{g(S)|X\}\} = E\{g(S)\}$, which was derived in (2.32), we obtain for the cross term in (6.6),

$$\begin{aligned}&E\big\{\big(S_n - E\{S_n \,|\, \mathcal{B}_n\}\,\big)\big(E\{S_n \,|\, \mathcal{B}_n\} - A_n(\mathcal{B}_n)\big)\big\} \\ &\qquad = E\big\{E\big\{\big(S_n - E\{S_n \,|\, \mathcal{B}_n\}\,\big)\big(E\{S_n \,|\, \mathcal{B}_n\} - A_n(\mathcal{B}_n)\big) \,|\, \mathcal{B}_n\big\}\big\} \\ &\qquad = E\{0\} = 0. \qquad (6.8)\end{aligned}$$

Inserting this relationship into (6.6) yields

$$\epsilon_U^2 = E\Big\{\big(S_n - E\{S_n \,|\, \mathcal{B}_n\}\,\big)^2\Big\} + \big(E\{S_n \,|\, \mathcal{B}_n\} - A_n(\mathcal{B}_n)\big)^2, \qquad (6.9)$$

which proves that the conditional mean $E\{S_n \,|\, \mathcal{B}_n\}$ minimizes the mean squared prediction error for a given observation set $\mathcal{B}_n$.

We will show later that in predictive coding the observation set $\mathcal{B}_n$ must consist of reconstructed samples. If we for example use the last N reconstructed samples as observation set, $\mathcal{B}_n = \{S'_{n-1}, \cdots, S'_{n-N}\}$, it is conceptually possible to construct a table in which the conditional expectations $E\big\{S_n \,|\, s'_{n-1}, \cdots, s'_{n-N}\big\}$ are stored for all possible combinations of the values of s'_{n-1} to s'_{n-N}. This is in some way similar to scalar quantization with an entropy coder that employs the conditional probabilities $p(s_n \,|\, s'_{n-1}, \cdots, s'_{n-N})$ and does not significantly reduce the complexity. For obtaining a low-complexity alternative to this scenario, we have to introduce structural constraints for the predictor $A_n(\mathcal{B}_n)$. Before we state a reasonable structural constraint, we derive the optimal predictors according to (6.5) for two examples.

Stationary Gaussian Sources. As a first example, we consider a stationary Gaussian source and derive the optimal predictor for a random variable S_n given a vector $\boldsymbol{S}_{n-k} = (S_{n-k}, \cdots, S_{n-k-N+1})^T$, with $k > 0$, of N preceding samples. The conditional distribution $f(S_n \,|\, \boldsymbol{S}_{n-k})$ of jointly Gaussian random variables is also Gaussian. The conditional mean $E\{S_n \,|\, \boldsymbol{S}_{n-k}\}$ and thus the optimal predictor is given by (see for example [26])

$$A_n(\boldsymbol{S}_{n-k}) = E\{S_n \,|\, \boldsymbol{S}_{n-k}\} = \mu_S + \boldsymbol{c}_k^T \, \boldsymbol{C}_N^{-1} \, (\boldsymbol{S}_{n-k} - \mu_S \, \boldsymbol{e}_N), \quad (6.10)$$

where μ_S represents the mean of the Gaussian process, $\boldsymbol{e}_N$ is the N-dimensional vector with all elements equal to 1, and $\boldsymbol{C}_N$ is the N-th order autocovariance matrix, which is given by

$$\boldsymbol{C}_N = E\big\{(\boldsymbol{S}_n - \mu_S \, \boldsymbol{e}_N)(\boldsymbol{S}_n - \mu_S \, \boldsymbol{e}_N)^T\big\} \,. \quad (6.11)$$

The vector $\boldsymbol{c}_k$ is an autocovariance vector and is given by

$$\boldsymbol{c}_k = E\{(S_n - \mu)(\boldsymbol{S}_{n-k} - \mu_S \, \boldsymbol{e}_N)\} \,. \quad (6.12)$$

Autoregressive Processes. Autoregressive processes are an important model for random sources. An autoregressive process of order m, also referred to as AR(m) process, is given by the recursive formula

$$\begin{aligned} S_n &= Z_n + \mu_S + \sum_{i=1}^{m} a_i \, (S_{n-1} - \mu_S) \\ &= Z_n + \mu_S(1 - \boldsymbol{a}_m^T \boldsymbol{e}_m) + \boldsymbol{a}_m^T \boldsymbol{S}_{n-1}^{(m)}, \end{aligned} \quad (6.13)$$

where μ_S is the mean of the random process, $\boldsymbol{a}_m = (a_1, \cdots, a_m)^T$ is a constant parameter vector, and $\{Z_n\}$ is a zero-mean iid process. We consider the prediction of a random variable S_n given the vector $\boldsymbol{S}_{n-1}$ of the N directly preceding samples, where N is greater than or equal to the order m. The optimal predictor is given by the conditional mean $E\{S_n \,|\, \boldsymbol{S}_{n-1}\}$. By defining an N-dimensional parameter vector $\boldsymbol{a}_N = (a_1, \cdots, a_m, 0, \cdots, 0)^T$, we obtain

$$\begin{aligned} E\{S_n \,|\, \boldsymbol{S}_{n-1}\} &= E\big\{Z_n + \mu_S(1 - \boldsymbol{a}_N^T \boldsymbol{e}_N) + \boldsymbol{a}_N^T \, \boldsymbol{S}_{n-1} \,|\, \boldsymbol{S}_{n-1}\big\} \\ &= \mu_S(1 - \boldsymbol{a}_N^T \boldsymbol{e}_N) + \boldsymbol{a}_N^T \, \boldsymbol{S}_{n-1}. \end{aligned} \quad (6.14)$$

For both considered examples, the optimal predictor is given by a linear function of the observation vector. In a strict sense, it is an affine function if the mean μ of the considered processes is nonzero. If we only want to minimize the variance of the prediction residual, we do not need the constant offset and can use strictly linear predictors. For predictive coding systems, affine predictors have the advantage that the scalar quantizer can be designed for zero-mean sources. Due to their simplicity and their effectiveness for a wide range of random processes, linear (and affine) predictors are the most important class of predictors for video coding applications. It should however be noted that nonlinear dependencies in the input process cannot be exploited using linear or affine predictors. In the following, we will concentrate on the investigation of linear prediction and linear predictive coding.

6.2 Linear Prediction

In the following, we consider linear and affine prediction of a random variable S_n given an observation vector $\boldsymbol{S}_{n-k} = [S_{n-k}, \cdots, S_{n-k-N+1}]^T$, with $k > 0$, of N preceding samples. We restrict our considerations to stationary processes. In this case, the prediction function $A_n(\boldsymbol{S}_{n-k})$ is independent of the time instant of the random variable to be predicted and is denoted by $A(\boldsymbol{S}_{n-k})$. For the more general affine form, the predictor is given by

$$\hat{S}_n = A(\boldsymbol{S}_{n-k}) = h_0 + \boldsymbol{h}_N^T \boldsymbol{S}_{n-k}, \tag{6.15}$$

where the constant vector $\boldsymbol{h}_N = (h_1, \cdots, h_N)^T$ and the constant offset h_0 are the parameters that characterize the predictor. For linear predictors, the constant offset h_0 is equal to zero.

The variance σ_U^2 of the prediction residual depends on the predictor parameters and can be written as

$$\begin{aligned}
\sigma_U^2(h_0, \boldsymbol{h}_N) &= E\Big\{\big(U_n - E\{U_n\}\big)^2\Big\} \\
&= E\Big\{\big(S_n - h_0 - \boldsymbol{h}_N^T \boldsymbol{S}_{n-k} - E\{S_n - h_0 - \boldsymbol{h}_N^T \boldsymbol{S}_{n-k}\}\big)^2\Big\} \\
&= E\Big\{\big(S_n - E\{S_n\} - \boldsymbol{h}_N^T\big(\boldsymbol{S}_{n-k} - E\{\boldsymbol{S}_{n-k}\}\big)\big)^2\Big\}.
\end{aligned} \tag{6.16}$$

The constant offset h_0 has no influence on the variance of the residual.

The variance σ_U^2 depends only on the parameter vector $\boldsymbol{h}_N$. By further reformulating the expression (6.16), we obtain

$$\begin{aligned}\sigma_U^2(\boldsymbol{h}_N) &= E\Big\{\big(S_n - E\{S_n\}\big)^2\Big\} \\ &\quad - 2\,\boldsymbol{h}_N^T\,E\big\{\big(S_n - E\{S_n\}\big)\big(\boldsymbol{S}_{n-k} - E\{\boldsymbol{S}_{n-k}\}\big)\big\} \\ &\quad + \boldsymbol{h}_N^T\,E\Big\{\big(\boldsymbol{S}_{n-k} - E\{\boldsymbol{S}_{n-k}\}\big)\big(\boldsymbol{S}_{n-k} - E\{\boldsymbol{S}_{n-k}\}\big)^T\Big\}\,\boldsymbol{h}_N \\ &= \sigma_S^2 - 2\,\boldsymbol{h}_N^T\,\boldsymbol{c}_k + \boldsymbol{h}_N^T\,\boldsymbol{C}_N\,\boldsymbol{h}_N, \end{aligned} \tag{6.17}$$

where σ_S^2 is the variance of the input process and $\boldsymbol{C}_N$ and $\boldsymbol{c}_k$ are the autocovariance matrix and the autocovariance vector of the input process given by (6.11) and (6.12), respectively.

The mean squared prediction error is given by

$$\begin{aligned}\epsilon_U^2(h_0, \boldsymbol{h}_N) &= \sigma_U^2(\boldsymbol{h}_N) + \mu_U^2(h_0, \boldsymbol{h}_N) \\ &= \sigma_U^2(\boldsymbol{h}_N) + E\big\{S_n - h_0 - \boldsymbol{h}_N^T\,\boldsymbol{S}_{n-k}\big\}^2 \\ &= \sigma_U^2(\boldsymbol{h}_N) + \big(\mu_S(1 - \boldsymbol{h}_N^T\boldsymbol{e}_N) - h_0\big)^2, \end{aligned} \tag{6.18}$$

with μ_S being the mean of the input process and $\boldsymbol{e}_N$ denoting the N-dimensional vector with all elements are equal to 1. Consequently, the minimization of the mean squared prediction error ϵ_U^2 is equivalent to choosing the parameter vector $\boldsymbol{h}_N$ that minimizes the variance σ_U^2 and additionally setting the constant offset h_0 equal to

$$h_0^* = \mu_S\,(1 - \boldsymbol{h}_N^T\,\boldsymbol{e}_N). \tag{6.19}$$

This selection of h_0 yields a mean of $\mu_U = 0$ for the prediction error signal, and the MSE between the input signal and the prediction ϵ_U^2 is equal to the variance of the prediction residual σ_U^2. Due to this simple relationship, we restrict the following considerations to linear predictors

$$\hat{S}_n = A(\boldsymbol{S}_{n-k}) = \boldsymbol{h}_N^T\,\boldsymbol{S}_{n-k} \tag{6.20}$$

and the minimization of the variance σ_U^2. But we keep in mind that the affine predictor that minimizes the mean squared prediction error can be obtained by additionally selecting an offset h_0 according to (6.19). The structure of a linear predictor is illustrated in Fig. 6.3.

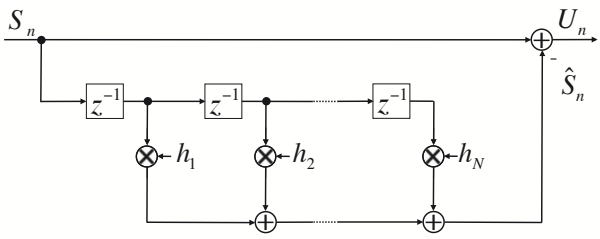


Fig. 6.3 Structure of a linear predictor.

6.3 Optimal Linear Prediction

A linear predictor is called an optimal linear predictor if its parameter vector $\boldsymbol{h}_N$ minimizes the variance $\sigma_U^2(\boldsymbol{h}_N)$ given in (6.17). The solution to this minimization problem can be obtained by setting the partial derivatives of σ_U^2 with respect to the parameters h_i, with $1 \leq i \leq N$, equal to 0. This yields the linear equation system

$$\boldsymbol{C}_N \, \boldsymbol{h}_N^* = \boldsymbol{c}_k. \tag{6.21}$$

We will prove later that this solution minimizes the variance σ_U^2. The N equations of the equation system (6.21) are also called the *normal equations* or the *Yule-Walker equations*. If the autocorrelation matrix $\boldsymbol{C}_N$ is nonsingular, the optimal parameter vector is given by

$$\boldsymbol{h}_N^* = \boldsymbol{C}_N^{-1} \, \boldsymbol{c}_k. \tag{6.22}$$

The autocorrelation matrix $\boldsymbol{C}_N$ of a stationary process is singular if and only if N successive random variables $S_n, S_{n+1}, \cdots, S_{n+N-1}$ are linearly dependent (see [75]), i.e., if the input process is deterministic. We ignore this case and assume that $\boldsymbol{C}_N$ is always nonsingular.

By inserting (6.22) into (6.17), we obtain the minimum prediction error variance

$$\begin{aligned}
\sigma_U^2(\boldsymbol{h}_N^*) &= \sigma_S^2 - 2\,(\boldsymbol{h}_N^*)^T \boldsymbol{c}_k + (\boldsymbol{h}_N^*)^T \boldsymbol{C}_N \, \boldsymbol{h}_N^* \\
&= \sigma_S^2 - 2\,\big(\boldsymbol{c}_k^T \boldsymbol{C}_N^{-1}\big)\boldsymbol{c}_k + \big(\boldsymbol{c}_k^T \boldsymbol{C}_N^{-1}\big)\boldsymbol{C}_N\big(\boldsymbol{C}_N^{-1}\boldsymbol{c}_k\big) \\
&= \sigma_S^2 - 2\,\boldsymbol{c}_k^T \boldsymbol{C}_N^{-1}\boldsymbol{c}_k + \boldsymbol{c}_k^T \boldsymbol{C}_N^{-1}\boldsymbol{c}_k \\
&= \sigma_S^2 - \boldsymbol{c}_k^T \boldsymbol{C}_N^{-1}\boldsymbol{c}_k.
\end{aligned} \tag{6.23}$$

Note that $(\boldsymbol{h}_N^*)^T = \boldsymbol{c}_k^T \, \boldsymbol{C}_N^{-1}$ follows from that fact that the autocorrelation matrix $\boldsymbol{C}_N$ and thus also its inverse $\boldsymbol{C}_N^{-1}$ is symmetric.

We now prove that the solution given by the normal equations (6.21) indeed minimizes the prediction error variance. Therefore, we investigate the prediction error variance for an arbitrary parameter vector $\boldsymbol{h}_N$, which can be represented as $\boldsymbol{h}_N = \boldsymbol{h}_N^* + \boldsymbol{\delta}_N$. Inserting this relationship into (6.17) and using (6.21) yields

$$\begin{aligned}
\sigma_U^2(\boldsymbol{h}_N) &= \sigma_S^2 - 2(\boldsymbol{h}_N^* + \boldsymbol{\delta}_N)^T \boldsymbol{c}_k + (\boldsymbol{h}_N^* + \boldsymbol{\delta}_N)^T \boldsymbol{C}_N\,(\boldsymbol{h}_N^* + \boldsymbol{\delta}_N) \\
&= \sigma_S^2 - 2\,(\boldsymbol{h}_N^*)^T \boldsymbol{c}_k - 2\,\boldsymbol{\delta}_N^T \boldsymbol{c}_k + \\
&\qquad (\boldsymbol{h}_N^*)^T \boldsymbol{C}_N\,\boldsymbol{h}_N^* + (\boldsymbol{h}_N^*)^T \boldsymbol{C}_N\,\boldsymbol{\delta}_N + \boldsymbol{\delta}_N^T \boldsymbol{C}_N\,\boldsymbol{h}_N^* + \boldsymbol{\delta}_N^T \boldsymbol{C}_N\,\boldsymbol{\delta}_N \\
&= \sigma_U^2(\boldsymbol{h}_N^*) - 2\boldsymbol{\delta}_N^T \boldsymbol{c}_k + 2\boldsymbol{\delta}_N^T \boldsymbol{C}_N\,\boldsymbol{h}_N^* + \boldsymbol{\delta}_N^T \boldsymbol{C}_n \boldsymbol{\delta}_N \\
&= \sigma_U^2(\boldsymbol{h}_N^*) + \boldsymbol{\delta}_N^T \boldsymbol{C}_N\,\boldsymbol{\delta}_N. \qquad (6.24)
\end{aligned}$$

It should be noted that the term $\boldsymbol{\delta}_N^T\,\boldsymbol{C}_N\,\boldsymbol{\delta}_N$ represents the variance $E\big\{(\boldsymbol{\delta}_N^T \boldsymbol{S}_n - E\big\{\boldsymbol{\delta}_N^T \boldsymbol{S}_n\big\})^2\big\}$ of the random variable $\boldsymbol{\delta}_N^T \boldsymbol{S}_n$ and is thus always greater than or equal to 0. Hence, we have

$$\sigma_U^2(\boldsymbol{h}_N) \geq \sigma_U^2(\boldsymbol{h}_N^*), \qquad (6.25)$$

which proves that (6.21) specifies the parameter vector $\boldsymbol{h}_N^*$ that minimizes the prediction error variance.

The Orthogonality Principle. In the following, we derive another important property for optimal linear predictors. We consider the more general affine predictor and investigate the correlation between the observation vector $\boldsymbol{S}_{n-k}$ and the prediction residual U_n,

$$\begin{aligned}
E\{U_n\,\boldsymbol{S}_{n-k}\} &= E\big\{\big(S_n - h_0 - \boldsymbol{h}_N^T \boldsymbol{S}_{n-k}\big)\,\boldsymbol{S}_{n-k}\big\} \\
&= E\{S_n\,\boldsymbol{S}_{n-k}\} - h_0\,E\{\boldsymbol{S}_{n-k}\} - E\big\{\boldsymbol{S}_{n-k}\boldsymbol{S}_{n-k}^T\big\}\,\boldsymbol{h}_N \\
&= \boldsymbol{c}_k + \mu_S^2\,\boldsymbol{e}_N - h_0\,\mu_S\,\boldsymbol{e}_N - (\boldsymbol{C}_N + \mu_S^2\,\boldsymbol{e}_N\,\boldsymbol{e}_N^T)\,\boldsymbol{h}_N \\
&= \boldsymbol{c}_k - \boldsymbol{C}_N \boldsymbol{h}_N + \mu_S\,\boldsymbol{e}_N\big(\mu_S\,(1 - \boldsymbol{h}_N^T\,\boldsymbol{e}_N) - h_0\big). \qquad (6.26)
\end{aligned}$$

By inserting the conditions (6.19) and (6.21) for optimal affine prediction, we obtain

$$E\{U_n\,\boldsymbol{S}_{n-k}\} = \boldsymbol{0}. \qquad (6.27)$$

Hence, optimal affine prediction yields a prediction residual U_n that is uncorrelated with the observation vector $\boldsymbol{S}_{n-k}$. For optimal linear

predictors, equation (6.27) holds only for zero-mean input signals. In general, only the covariance between the prediction residual and each observation is equal to zero,

$$E\big\{\big(U_n - E\{U_n\}\big)\big(\boldsymbol{S}_{n-k} - E\{\boldsymbol{S}_{n-k}\}\big)\big\} = \boldsymbol{0}. \tag{6.28}$$

Prediction of Vectors. The linear prediction for a single random variable S_n given an observation vector $\boldsymbol{S}_{n-k}$ can also be extended to the prediction of a vector $\boldsymbol{S}_{n+K-1} = (S_{n+K-1}, S_{n+K-2}, \cdots, S_n)^T$ of K random variables. For each random variable of $\boldsymbol{S}_{n+K-1}$, the optimal linear or affine predictor can be derived as discussed above. If the parameter vectors $\boldsymbol{h}_N$ are arranged in a matrix and the offsets h_0 are arranged in a vector, the prediction can be written as

$$\hat{\boldsymbol{S}}_{n+K-1} = \boldsymbol{H}_K \cdot \boldsymbol{S}_{n-k} + \boldsymbol{h}_K, \tag{6.29}$$

where $\boldsymbol{H}_K$ is an $K \times N$ matrix whose rows are given by the corresponding parameter vectors $\boldsymbol{h}_N$ and $\boldsymbol{h}_K$ is a K-dimensional vector whose elements are given by the corresponding offsets h_0.

6.3.1 One-Step Prediction

The most often used prediction is the one-step prediction in which a random variable S_n is predicted using the N directly preceding random variables $\boldsymbol{S}_{n-1} = (S_{n-1}, \cdots, S_{n-N})^T$. For this case, we now derive some useful expressions for the minimum prediction error variance $\sigma_U^2(\boldsymbol{h}_N^*)$, which will be used later for deriving an asymptotic bound.

For the one-step prediction, the normal equations (6.21) can be written in matrix notation as

$$\begin{bmatrix} \phi_0 & \phi_1 & \cdots & \phi_{N-1} \\ \phi_1 & \phi_0 & \cdots & \phi_{N-2} \\ \vdots & \vdots & \ddots & \vdots \\ \phi_{N-1} & \phi_{N-2} & \cdots & \phi_0 \end{bmatrix} \begin{bmatrix} h_1^N \\ h_2^N \\ \vdots \\ h_N^N \end{bmatrix} = \begin{bmatrix} \phi_1 \\ \phi_2 \\ \vdots \\ \phi_N \end{bmatrix}, \tag{6.30}$$

where the factors h_k^N represent the elements of the optimal parameter vector $\boldsymbol{h}_N^* = (h_1^N, \cdots, h_N^N)^T$ for linear prediction using the N preceding samples. The covariances $E\big\{\big(S_n - E\{S_n\}\big)\big(S_{n+k} - E\{S_{n+k}\}\big)\big\}$ are denoted by ϕ_k. By adding a matrix column to the left, multiplying

the parameter vector $\boldsymbol{h}_N^*$ with -1, and adding an element equal to 1 at the top of the parameter vector, we obtain

$$\begin{bmatrix} \phi_1 & \phi_0 & \phi_1 & \cdots & \phi_{N-1} \\ \phi_2 & \phi_1 & \phi_0 & \cdots & \phi_{N-2} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ \phi_N & \phi_{N-1} & \phi_{N-2} & \cdots & \phi_0 \end{bmatrix} \begin{bmatrix} 1 \\ -h_1^N \\ -h_2^N \\ \vdots \\ -h_N^N \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ \vdots \\ 0 \end{bmatrix}. \tag{6.31}$$

We now include the expression for the minimum prediction variance into the matrix equation. The prediction error variance for optimal linear prediction using the N preceding samples is denoted by σ_N^2. Using (6.23) and (6.22), we obtain

$$\sigma_N^2 = \sigma_S^2 - \boldsymbol{c}_1^T \boldsymbol{h}_N^* = \phi_0 - h_1^N \phi_1 - h_2^N \phi_2 - \cdots - h_N^N \phi_N. \tag{6.32}$$

Adding this relationship to the matrix equation (6.31) yields

$$\begin{bmatrix} \phi_0 & \phi_1 & \phi_2 & \cdots & \phi_N \\ \phi_1 & \phi_0 & \phi_1 & \cdots & \phi_{N-1} \\ \phi_2 & \phi_1 & \phi_0 & \cdots & \phi_{N-2} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ \phi_N & \phi_{N-1} & \phi_{N-2} & \cdots & \phi_0 \end{bmatrix} \begin{bmatrix} 1 \\ -h_1^N \\ -h_2^N \\ \vdots \\ -h_N^N \end{bmatrix} = \begin{bmatrix} \sigma_N^2 \\ 0 \\ 0 \\ \vdots \\ 0 \end{bmatrix}. \tag{6.33}$$

This equation is also referred to as the *augmented normal equation*. It should be noted that the matrix on the left represents the autocovariance matrix $\boldsymbol{C}_{N+1}$. We denote the modified parameter vector by $\boldsymbol{a}_N = (1, -h_1^N, \cdots, -h_N^N)^T$. By multiplying both sides of (6.33) from the left with the transpose of $\boldsymbol{a}_N$, we obtain

$$\sigma_N^2 = \boldsymbol{a}_N^T \, \boldsymbol{C}_{N+1} \, \boldsymbol{a}_N. \tag{6.34}$$

We have one augmented normal equation (6.33) for each particular number N of preceding samples in the observation vector. Combing the equations for 0 to N preceding samples into one matrix equation yields

$$\boldsymbol{C}_{N+1} \cdot \begin{bmatrix} 1 & 0 & \cdots & 0 & 0 \\ -h_1^N & 1 & \ddots & 0 & 0 \\ -h_2^N & -h_1^{N-1} & \ddots & 0 & 0 \\ \vdots & \vdots & \ddots & 1 & 0 \\ -h_N^N & -h_{N-1}^{N-1} & \cdots & -h_1^1 & 1 \end{bmatrix} = \begin{bmatrix} \sigma_N^2 & X & \cdots & X & X \\ 0 & \sigma_{N-1}^2 & \ddots & X & X \\ 0 & 0 & \ddots & X & X \\ \vdots & \vdots & \ddots & \sigma_1^2 & X \\ 0 & 0 & 0 & 0 & \sigma_0^2 \end{bmatrix}, \tag{6.35}$$

where X represents arbitrary values and σ_0^2 is the variance of the input signal. Taking the determinant of both sides of the equation gives

$$|\boldsymbol{C}_{N+1}| = \sigma_N^2 \cdot \sigma_{N-1}^2 \cdot \ldots \cdot \sigma_0^2. \tag{6.36}$$

Note that the determinant of a triangular matrix is the product of the elements on its main diagonal. Hence, the prediction error variance σ_N^2 for optimal linear prediction using the N preceding samples can also be written as

$$\sigma_N^2 = \frac{|\boldsymbol{C}_{N+1}|}{|\boldsymbol{C}_N|}. \tag{6.37}$$

6.3.2 One-Step Prediction for Autoregressive Processes

In the following, we consider the particularly interesting case of optimal linear one-step prediction for autoregressive processes. As stated in sec. 6.1, an AR(m) process with the mean μ_S is defined by

$$S_n = Z_n + \mu_S(1 - \boldsymbol{a}_m^T \boldsymbol{e}_m) + \boldsymbol{a}_m^T \boldsymbol{S}_{n-1}^{(m)}, \tag{6.38}$$

where $\{Z_n\}$ is a zero-mean iid process and $\boldsymbol{a}_m = (a_1, \cdots, a_m)^T$ is a constant parameter vector. We consider the one-step prediction using the N preceding samples and the prediction parameter vector $\boldsymbol{h}_N$. We assume that the number N of preceding samples in the observation vector $\boldsymbol{S}_{n-1}$ is greater than or equal to the process order m and define a vector $\boldsymbol{a}_N = (a_1, \cdots, a_m, 0, \cdots, 0)^T$ whose first m elements are given by the process parameter vector $\boldsymbol{a}_m$ and whose last $N - m$ elements are equal to 0. The prediction residual can then be written as

$$U_n = Z_n + \mu_S(1 - \boldsymbol{a}_N^T \boldsymbol{e}_N) + (\boldsymbol{a}_N - \boldsymbol{h}_N)^T \boldsymbol{S}_{n-1}. \tag{6.39}$$

By subtracting the mean $E\{U_n\}$ we obtain

$$U_n - E\{U_n\} = Z_n + (\boldsymbol{a}_N - \boldsymbol{h}_N)^T \big(\boldsymbol{S}_{n-1} - E\{\boldsymbol{S}_{n-1}\}\big). \tag{6.40}$$

According to (6.28), the covariances between the residual U_n and the random variables of the observation vector must be equal to 0 for optimal linear prediction. This gives

$$\begin{aligned} \boldsymbol{0} &= E\big\{\big(U_n - E\{U_n\}\big)\big(\boldsymbol{S}_{n-k} - E\{\boldsymbol{S}_{n-k}\}\big)\big\} \\ &= E\big\{Z_n\big(\boldsymbol{S}_{n-k} - E\{\boldsymbol{S}_{n-k}\}\big)\big\} + \boldsymbol{C}_N(\boldsymbol{a}_N - \boldsymbol{h}_N). \end{aligned} \tag{6.41}$$

Since $\{Z_n\}$ is an iid process, Z_n is independent of the past $\boldsymbol{S}_{n-k}$, and the expectation value in (6.41) is equal to 0. The optimal linear predictor is given by

$$\boldsymbol{h}_N^* = \boldsymbol{a}_N. \tag{6.42}$$

Hence, for AR(m) processes, optimal linear prediction can be achieved by using the m preceding samples as observation vector and setting the prediction parameter vector $\boldsymbol{h}_m$ equal to the parameter vector $\boldsymbol{a}_m$ of the AR(m) process. An increase of the prediction order N does not result in a decrease of the prediction error variance. All prediction parameters h_k with $k > m$ are equal to 0. It should be noted that if the prediction order N is less than the process order m, the optimal prediction coefficients h_k are in general not equal to the corresponding process parameters a_k. In that case, the optimal prediction vector must be determined according to the normal equations (6.21).

If the prediction order N is greater than or equal to the process order m, the prediction residual becomes

$$U_n = Z_n + \mu_U \qquad \text{with} \qquad \mu_U = \mu_S(1 - \boldsymbol{a}_m^T \boldsymbol{e}_m). \tag{6.43}$$

The prediction residual is an iid process. Consequently, optimal linear prediction of AR(m) processes with a prediction order N greater than or equal to the process order m yields an iid residual process $\{U_n\}$ (white noise) with a mean μ_U and a variance $\sigma_U^2 = E\{Z_n^2\}$.

Gauss-Markov Processes. A Gauss-Markov process is a particular AR(1) process,

$$S_n = Z_n + \mu_S(1 - \rho) + \rho \cdot S_{n-1}, \tag{6.44}$$

for which the iid process $\{Z_n\}$ has a Gaussian distribution. It is completely characterized by its mean μ_S, its variance σ_S^2, and the correlation coefficient ρ with $-1 < \rho < 1$. According to the analysis above, the optimal linear predictor for Gauss-Markov processes consists of a single coefficient h_1 that is equal to ρ. The obtained prediction residual process $\{U_n\}$ represents white Gaussian noise with a mean $\mu_U = \mu_S(1-\rho)$ and a variance

$$\sigma_U^2 = \frac{|\boldsymbol{C}_2|}{|\boldsymbol{C}_1|} = \frac{\sigma_S^4 - \sigma_S^4\,\rho^2}{\sigma_S^2} = \sigma_S^2\,(1 - \rho^2). \tag{6.45}$$

6.3.3 Prediction Gain

For measuring the effectiveness of a prediction often the *prediction gain* G_P is used, which can be defined as the ratio of the signal variance and the variance of the prediction residual,

$$G_P = \frac{\sigma_S^2}{\sigma_U^2}. \tag{6.46}$$

For a fixed prediction structure, the prediction gain for optimal linear prediction does only depend on the autocovariances of the sources process. The prediction gain for optimal linear one-step prediction using the N preceding samples is given by

$$G_P = \frac{\sigma_S^2}{\sigma_S^2 - \boldsymbol{c}_1^T \boldsymbol{C}_N \boldsymbol{c}_1} = \frac{1}{1 - \boldsymbol{\phi}_1^T \boldsymbol{\Phi}_N \boldsymbol{\phi}_1}, \tag{6.47}$$

where $\boldsymbol{\Phi}_N = \boldsymbol{C}_N/\sigma_S^2$ and $\boldsymbol{\phi}_i = \boldsymbol{c}_1/\sigma_S^2$ are the normalized autocovariance matrix and the normalized autocovariance vector, respectively.

The prediction gain for the one-step prediction of Gauss-Markov processes with a prediction coefficient h_1 is given by

$$G_P = \frac{\sigma_S^2}{\sigma_S^2 - 2h_1\sigma_S^2\rho + h_1^2\sigma_S^2} = \frac{1}{1 - 2h_1\rho + h_1^2}. \tag{6.48}$$

For optimal linear one-step prediction ($h_1 = \rho$), we obtain

$$G_P = \frac{1}{1 - \rho^2}. \tag{6.49}$$

For demonstrating the impact of choosing the prediction coefficient h_1 for the linear one-step prediction of Gauss-Markov sources, Fig. 6.4 shows the prediction error variance and the prediction gain for a linear predictor with a fixed prediction coefficient of $h_1 = 0.5$ and for the optimal linear predictor ($h_1 = \rho$) as function of the correlation factor ρ.

6.3.4 Asymptotic Prediction Gain

In the previous sections, we have focused on linear and affine prediction with a fixed-length observation vector. Theoretically, we can make the prediction order N very large and for N approaching infinity we obtain

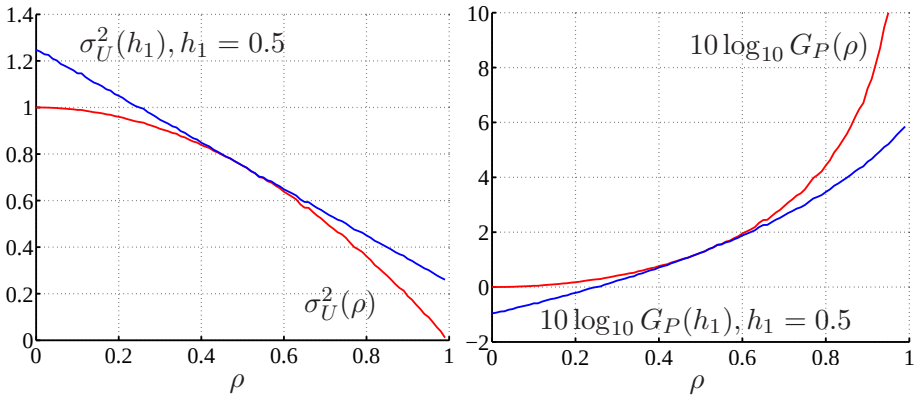


Fig. 6.4 Linear one-step prediction for Gauss-Markov processes with unit variance. The diagrams show the prediction error variance (left) and the prediction gain (right) for a linear predictor with $h_1 = 0.5$ (blues curve) and an optimal linear predictor with $h_1 = \rho$ (red curves) in dependence of the correlation factor ρ.

an upper bound for the prediction gain. For deriving this bound, we consider the one-step prediction of a random variable S_n given the countably infinite set of preceding random variables $\{S_{n-1}, S_{n-2}, \cdots\}$. For affine prediction, the prediction residual can be written as

$$U_n = S_n - h_0 - \sum_{i=1}^{\infty} h_i \, S_{n-i}, \tag{6.50}$$

where the set $\{h_0, h_1, \cdots\}$ is a countably infinite set of prediction coefficients. According to the orthogonality condition (6.27), the prediction residual U_n is uncorrelated with all preceding random variables S_{n-k} with $k > 0$. In addition, each prediction residual U_{n-k} with $k > 0$ is completely determined by a linear combination (6.50) of the random variables S_{n-k-i} with $i \geq 0$. Consequently, U_n is also uncorrelated with the preceding prediction residuals U_{n-k} with $k > 0$. Hence, if the prediction order N approaches infinity, the generated sequence of prediction residuals $\{U_n\}$ represents an uncorrelated sequence. Its power spectral density is given by

$$\Phi_{UU}(\omega) = \sigma_{U,\infty}^2, \tag{6.51}$$

where $\sigma_{U,\infty}^2$ denotes the asymptotic one-step prediction error variance for N approaching infinity.

For deriving an expression for the asymptotic one-step prediction error variance $\sigma^2_{U,\infty}$, we restrict our considerations to zero-mean input processes, for which the autocovariance matrix $\boldsymbol{C}_N$ is equal to the corresponding autocorrelation matrix $\boldsymbol{R}_N$, and first consider the limit

$$\lim_{N\to\infty} |\boldsymbol{C}_N|^{\frac{1}{N}}. \tag{6.52}$$

Since the determinant of a $N\times N$ matrix is given by the product of its eigenvalues $\xi_i^{(N)}$, with $i = 0, 1, \cdots, N-1$, we can write

$$\lim_{N\to\infty} |\boldsymbol{C}_N|^{\frac{1}{N}} = \lim_{N\to\infty} \left(\prod_{i=0}^{N-1} \xi_i^{(N)} \right)^{\frac{1}{N}} = 2^{\left(\lim\limits_{N\to\infty} \sum\limits_{i=0}^{N-1} \frac{1}{N} \log_2 \xi_i^{(N)} \right)}. \tag{6.53}$$

By applying Grenander and Szegö's theorem for sequences of Toeplitz matrices (4.76), we obtain

$$\lim_{N\to\infty} |\boldsymbol{C}_N|^{\frac{1}{N}} = 2^{\frac{1}{2\pi}\int_{-\pi}^{\pi} \log_2 \Phi_{SS}(\omega)\,\mathrm{d}\omega}, \tag{6.54}$$

where $\Phi_{SS}(\omega)$ denotes the power spectral density of the input process $\{S_n\}$. As a further consequence of the convergence of the limit in (6.52), we can state

$$\lim_{N\to\infty} \frac{|\boldsymbol{C}_{N+1}|^{\frac{1}{N+1}}}{|\boldsymbol{C}_N|^{\frac{1}{N}}} = 1. \tag{6.55}$$

According to (6.37), we can express the asymptotic one-step prediction error variance $\sigma^2_{U,\infty}$ by

$$\sigma^2_{U,\infty} = \lim_{N\to\infty} \frac{|\boldsymbol{C}_{N+1}|}{|\boldsymbol{C}_N|} = \lim_{N\to\infty} \left(\frac{|\boldsymbol{C}_{N+1}|^{\frac{1}{N+1}}}{|\boldsymbol{C}_N|^{\frac{1}{N}}\ |\boldsymbol{C}_N|^{-\frac{1}{N(N+1)}}} \right)^{N+1}. \tag{6.56}$$

Applying (6.54) and (6.55) yields

$$\sigma^2_{U,\infty} = \lim_{N\to\infty} |\boldsymbol{C}_N|^{\frac{1}{N}} = 2^{\frac{1}{2\pi}\int_{-\pi}^{\pi} \log_2 \Phi_{SS}(\omega)\,\mathrm{d}\omega}. \tag{6.57}$$

Hence, the asymptotic linear prediction gain for zero-mean input sources is given by

$$G_P^{\infty} = \frac{\sigma_S^2}{\sigma^2_{U,\infty}} = \frac{\frac{1}{2\pi}\int_{-\pi}^{\pi} \Phi_{SS}(\omega) d\omega}{2^{\frac{1}{2\pi}\int_{-\pi}^{\pi} \log_2 \Phi_{SS}(\omega) d\omega}}. \tag{6.58}$$

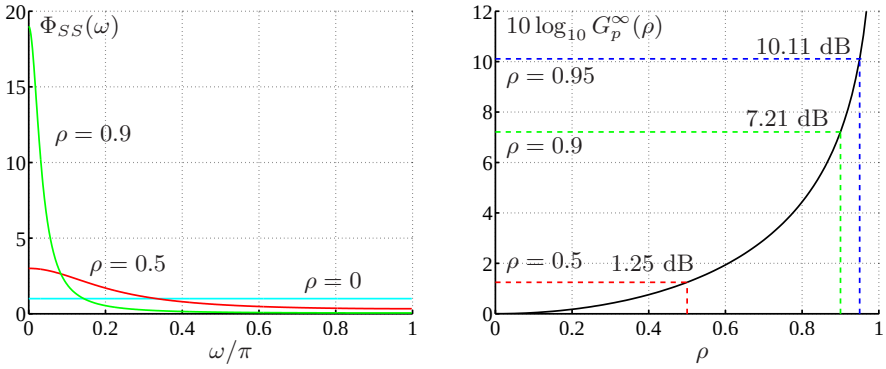


Fig. 6.5 Prediction gain for zero-mean Gauss-Markov sources: (left) Power spectral density; (right) Prediction gain.

It should be noted that for zero-mean AR(m) processes, such as zero-mean Gauss-Markov processes, this asymptotic prediction gain is already achieved by using optimal linear one-step predictors of a finite order $N \geq m$. As an example, we know from (4.77) to (4.79) that

$$\frac{1}{2\pi} \int_{-\pi}^{\pi} \log_2 \Phi_{SS}(\omega) \, \mathrm{d}\omega = \log_2\big(\sigma_S^2 \, (1 - \rho^2)\big) \tag{6.59}$$

for Gauss-Markov processes. This yields the asymptotic prediction gain $G_P^{\infty} = 1/(1 - \rho^2)$, which we have already derived for the optimal one-step prediction in (6.45). This relationship can also be obtained by inserting the expression (2.50) for the determinant $|\boldsymbol{C}_N|$ into (6.57). Fig. 6.5 illustrates the power spectral density and the prediction gain for stationary zero-mean Gauss-Markov processes.

6.4 Differential Pulse Code Modulation (DPCM)

In the previous sections, we investigated the prediction and in particular the linear prediction of a random variable S_n using the values of preceding random variables. We now consider the combination of prediction and scalar quantization.

We first consider the case that the random variables of the input process are predicted as discussed in the previous sections (i.e., using the original values of preceding samples) and the resulting prediction residuals are quantized. For the example of one-step prediction using

the directly preceding sample, we obtain the encoder reconstructions

$$S'_{n,e} = U'_n + \hat{S}_{n,e} = Q(S_n - A(S_{n-1})) + A(S_{n-1}). \tag{6.60}$$

At the decoder side, however, we do not know the original sample values. Here we must use the reconstructed values for deriving the prediction values. The corresponding decoder reconstructions are given by

$$S'_{n,d} = U'_n + \hat{S}_{n,d} = Q(S_n - A(S_{n-1})) + A(S'_{n-1,d}). \tag{6.61}$$

For such an *open-loop predictive coding* structure, the encoder and decoder reconstructions $S'_{n,e}$ and $S'_{n,d}$ differ by $P(S_{n-1}) - P(S'_{n-1,d})$. If we use a recursive prediction structure as in the considered one-step prediction, the differences between encoder and decoder reconstructions increase over time. This effect is also referred to as *drift* and can only be avoided if the prediction at both encoder and decoder side uses reconstructed samples.

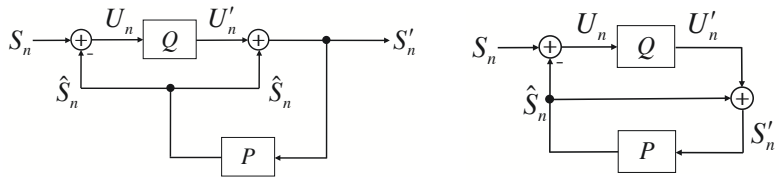


Fig. 6.6 Closed-loop predictive coding: (left) Prediction structure using reconstructed samples for forming the prediction signal; (right) DPCM structure.

The basic structure of a predictor that uses reconstructed samples S'_n for forming the prediction signal is shown in the left block diagram of Fig. 6.6. This structure is also referred to as *closed-loop predictive coding* structure and is used in basically all video coding applications. The closed-loop structure ensures that a decoder can obtain the same reconstruction values as the encoder. By redrawing the block diagram without changing the signal flow we obtain the structure shown in the right block diagram of Fig. 6.6, which is also referred to as *differential pulse code modulation* (DPCM).

If we decompose the quantizer Q in Fig. 6.6 into an encoder mapping α that maps the prediction residuals U_n onto quantization indexes I_n and a decoding mapping β that maps the quantization indexes I_n onto reconstructed residuals U'_n and add a lossless coding γ

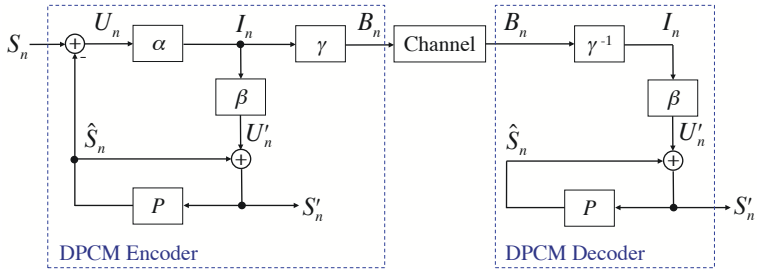


Fig. 6.7 Block diagram of a DPCM encoder and decoder.

for mapping the quantization indexes I_n onto codewords B_n, we obtain the well-known structure of a DPCM encoder shown on the left side of Fig. 6.7. The corresponding DPCM decoder is shown on the right side of Fig. 6.7. It includes, the inverse lossless coding γ^{-1}, the decoder mapping β, and the predictor. If the codewords are transmitted over an error-free channel, the reconstruction values at the decoder side are identical to the reconstruction values at the encoder side, since the mapping of the quantization indexes I_n to reconstructed values S'_n is the same in both encoder and decoder. The DPCM encoder contains the DPCM decoder except for the inverse lossless coding γ^{-1}.

6.4.1 Linear Prediction for DPCM

In sec. 6.3, we investigated optimal linear prediction of a random variable S_n using original sample values of the past. However, in DPCM coding, the prediction $\hat{S}_n$ for a random variable S_n must be generated by a linear combination of the reconstructed values S'_n of already coded samples. If we consider linear one-step prediction using an observation vector $\boldsymbol{S'}_{n-1} = (S'_{n-1}, \cdots, S'_{n-N})^T$ that consists of the reconstruction values of the N directly preceding samples, the prediction value $\hat{S}_n$ can be written as

$$\hat{S}_n = \sum_{i=1}^{N} h_i \; S'_{n-i} = \sum_{i=1}^{K} h_i \; (S_{n-i} + Q_{n-i}) = \boldsymbol{h}_N^T \, (\boldsymbol{S}_{n-1}^N + \boldsymbol{Q}_{n-1}^N), \quad (6.62)$$

where $Q_n = U'_n - U_n$ denotes the quantization error, $\boldsymbol{h}_N$ is the vector of prediction parameters, $\boldsymbol{S}_{n-1} = (S_{n-1}, \cdots, S_{n-N})^T$ is the vector of the N original sample values that precede the current sample S_n to be

predicted, and $\boldsymbol{Q}_{n-1} = (Q_{n-1}, \cdots, Q_{n-N})^T$ is the vector of the quantization errors for the N preceding samples. The variance σ_U^2 of the prediction residual U_n is given by

$$\begin{aligned}
\sigma_U^2 &= E\big\{(U_n - E\{U_n\})^2\big\} \\
&= E\Big\{\big(S_n - E\{S_n\} - \boldsymbol{h}_N^T\,\big(\boldsymbol{S}_{n-1} - E\{\boldsymbol{S}_{n-1}\} + \boldsymbol{Q}_{n-1} - E\big\{\boldsymbol{Q}_{n-1}\big\}\,\big)\big)^2\Big\} \\
&= \sigma_S^2 - 2\,\boldsymbol{h}_N^T\,\boldsymbol{c}_1 + \boldsymbol{h}_N^T\,\boldsymbol{C}_N\,\boldsymbol{h}_N \\
&\quad - 2\,\boldsymbol{h}_N^T\,E\big\{\big(S_n - E\{S_n\}\big)\big(\boldsymbol{Q}_{n-1} - E\big\{\boldsymbol{Q}_{n-1}\big\}\big)\big\} \\
&\quad - 2\,\boldsymbol{h}_N^T\,E\Big\{\big(\boldsymbol{S}_{n-1} - E\{\boldsymbol{S}_{n-1}\}\big)\big(\boldsymbol{Q}_{n-1} - E\big\{\boldsymbol{Q}_{n-1}\big\}\big)^T\Big\}\,\boldsymbol{h}_N \\
&\quad + \boldsymbol{h}_N^T\,E\Big\{\big(\boldsymbol{Q}_{n-1} - E\big\{\boldsymbol{Q}_{n-1}\big\}\big)\big(\boldsymbol{Q}_{n-1} - E\big\{\boldsymbol{Q}_{n-1}\big\}\big)^T\Big\}\,\boldsymbol{h}_N. \qquad (6.63)
\end{aligned}$$

The optimal prediction parameter vector $\boldsymbol{h}_N$ does not only depend on the autocovariances of the input process $\{S_n\}$, but also on the autocovariances of the quantization errors $\{Q_n\}$ and the cross-covariances between the input process and the quantization errors. Thus, we need to know the quantizer in order to design an optimal linear predictor. But on the other hand, we also need to know the predictor parameters for designing the quantizer. Thus, for designing a optimal DPCM coder the predictor and quantizer have to be optimized jointly. Numerical algorithms that iteratively optimize the predictor and quantizer based on conjugate gradient numerical techniques are discussed in [8].

For high rates, the reconstructed samples S'_n are a close approximation of the original samples S_n, and the optimal prediction parameter vector $\boldsymbol{h}_N$ for linear prediction using reconstructed sample values is virtually identical to the optimal prediction parameter vector for linear prediction using original sample values. In the following, we concentrate on DPCM systems for which the linear prediction parameter vector is optimized for a prediction using original sample values, but we note that such DPCM systems are suboptimal for low rates.

One-Tap Prediction for Gauss-Markov Sources. As an important example, we investigate the rate distortion efficiency of linear predictive coding for stationary Gauss-Markov sources,

$$S_n = Z_n + \mu_S\,(1 - \rho) + \rho\,S_{n-1}. \qquad (6.64)$$

We have shown in sec. 6.3.2 that the optimal linear predictor using original sample values is the one-tap predictor for which the prediction coefficient h_1 equal to the correlation coefficient ρ of the Gauss-Markov process. If we use the same linear predictor with reconstructed samples, the prediction $\hat{S}_n$ for a random variable S_n can be written as

$$\hat{S}_n = h_1 \, S'_{n-1} = \rho \, (S_{n-1} + Q_{n-1}), \tag{6.65}$$

where $Q_{n-1} = U'_{n-1} - U_{n-1}$ denotes the quantization error. The prediction residual U_n is given by

$$U_n = S_n - \hat{S}_n = Z_n + \mu_S \, (1 - \rho) - \rho \, Q_{n-1}. \tag{6.66}$$

For the prediction error variance σ_U^2, we obtain

$$\begin{aligned} \sigma_U^2 &= E\big\{(U_n - E\{U_n\})^2\big\} = E\Big\{\big(Z_n - \rho\,(Q_{n-1} - E\{Q_{n-1}\})\big)^2\Big\} \\ &= \sigma_Z^2 - 2\,\rho\,E\{Z_n\,(Q_{n-1} - E\{Q_{n-1}\})\} + \rho^2\,\sigma_Q^2, \end{aligned} \tag{6.67}$$

where $\sigma_Z^2 = E\big\{Z_n^2\big\}$ denotes the variance of the innovation process $\{Z_n\}$ and $\sigma_Q^2 = E\big\{(Q_n - E\{Q_n\})^2\big\}$ denotes the variance of the quantization errors. Since $\{Z_n\}$ is an iid process and thus Z_n is independent of the past quantization errors Q_{n-1}, the middle term in (6.67) is equal to 0. Furthermore, as shown in sec. 2.3.1, the variance σ_Z^2 of the innovation process is given by $\sigma_S^2\,(1 - \rho^2)$. Hence, we obtain

$$\sigma_U^2 = \sigma_S^2\,(1 - \rho^2) + \rho^2\,\sigma_Q^2. \tag{6.68}$$

We further note that the quantization error variance σ_Q^2 represents the distortion D of the DPCM quantizer and is a function of the rate R. As explained in sec. 5.2.4, we can generally express the distortion rate function of scalar quantizers by

$$D(R) = \sigma_Q^2(R) = \sigma_U^2(R)\;g(R), \tag{6.69}$$

where $\sigma_U^2(R)$ represents the variance of the signal that is quantized. The function $g(R)$ represents the operational distortion rate function for quantizing random variables that have the same distribution type as the prediction residual U_n, but unit variance. Consequently, the variance of the prediction residual is given by

$$\sigma_U^2(R) = \sigma_S^2\,\frac{1 - \rho^2}{1 - \rho^2\,g(R)}. \tag{6.70}$$

Using (6.69), we obtain the following operational distortion rate function for linear predictive coding of Gauss-Markov processes with a one-tap predictor for which the prediction coefficient h_1 is equal to the correlation coefficient of the Gauss-Markov source,

$$D(R) = \sigma_S^2 \, \frac{1-\rho^2}{1-\rho^2 \, g(R)} \, g(R). \tag{6.71}$$

By deriving the asymptote for $g(R)$ approaching zero, we obtain the following asymptotic operational distortion rate function for high rates,

$$D(R) = \sigma_S^2 \, (1-\rho^2) \, g(R). \tag{6.72}$$

The function $g(R)$ represents the operational distortion rate function for scalar quantization of random variables that have unit variance and the same distribution type as the prediction residuals. It should be mentioned that, even at high rates, the distribution of the prediction residuals cannot be derived in a straightforward way, since it is determined by a complicated process that includes linear prediction and quantization. As a rule of thumb based on intuition, at high rates, the reconstructed values S'_n are a very close approximation of the original samples S_n and thus the quantization errors $Q_n = S'_n - S_n$ are very small in comparison to the innovation Z_n. Then, we can argue that the prediction residuals U_n given by (6.66) are nearly identical to the innovation samples Z_n and have thus nearly a Gaussian distribution. Another reason for assuming a Gaussian model is the fact that Gaussian sources are the most difficult to code among all processes with a given autocovariance function. Using a Gaussian model for the prediction residuals, we can replace $g(R)$ in (6.72) by the high rate asymptote for entropy-constrained quantization of Gaussian sources, which yields the following high rate approximation of the operational distortion rate function,

$$D(R) = \frac{\pi \, e}{6} \, \sigma_S^2 \, (1-\rho^2) \, 2^{-2R}. \tag{6.73}$$

Hence, under the intuitive assumption that the distribution of the prediction residuals at high rates is nearly Gaussian, we obtain an asymptotic operational distortion rate function for DPCM quantization of stationary Gauss-Markov processes at high rates that lies

1.53 dB or 0.25 bit per sample above the fundamental rate distortion bound (4.119). The experimental results presented below indicate that our intuitive assumption provides a useful approximation of the operational distortion rate function for DPCM coding of stationary Gauss-Markov processes at high rates.

Entropy-Constrained Lloyd Algorithm for DPCM. Even if we use the optimal linear predictor for original sample values inside the DPCM loop, the quantizer design algorithm is not straightforward, since the distribution of the prediction residuals depends on the reconstructed sample values and thus on the quantizer itself.

In order to provide some experimental results for DPCM quantization of Gauss-Markov sources, we use a very simple ECSQ design in combination with a given linear predictor. The vector of prediction parameters $\boldsymbol{h}_N$ is given and only the entropy-constrained scalar quantizer is designed. Given a sufficiently large training set $\{s_n\}$, the quantizer design algorithm can be stated as follows:

(1) Initialized the Lagrange multiplier λ with small value and initialize all reconstructed samples s'_n with the corresponding original samples s_n of the training set.
(2) Generate the residual samples using linear prediction given the original and reconstructed samples s_n and s'_n.
(3) Design an entropy-constrained Lloyd quantizer as described in sec. 5.2.2 given the value of λ and using the prediction error sequence $\{u_n\}$ as training set.
(4) Conduct the DPCM coding of the training set $\{s_n\}$ given the linear predictor and the designed quantizer, which yields the set of reconstructed samples $\{s'_n\}$.
(5) Increase λ by a small amount and start again with step 2.

The quantizer design algorithm starts with a small value of λ and thus a high rate for which we can assume that reconstruction values are nearly identical to the original sample values. In each iteration of the algorithm, a quantizer is designed for a slightly larger value of λ and thus a slightly lower rate by assuming that the optimal quantizer

design does not change significantly. By executing the algorithm, we obtain a sequence of quantizers for different rates. It should however be noted that the quantizer design inside a feedback loop is a complicated problem. We noted that when the value of λ is changed too much from one iteration to the next, the algorithm becomes unstable at low rates. An alternative algorithm for designing predictive quantizers based on conjugate gradient techniques can be found in [8].

Experimental Results for a Gauss-Markov Source. For providing experimental results, we considered the stationary Gauss-Markov source with zero mean, unit variance, and a correlation factor of 0.9 that we have used as reference throughout this text. We have run the entropy-constrained Lloyd algorithm for DPCM stated above and measured the prediction error variance σ_U^2, the distortion D, and the entropy of the reconstructed sample values as measure for the transmission rate R. The results of the algorithm are compared to the distortion rate function and to the derived functions for $\sigma_U^2(R)$ and $D(R)$ for stationary Gauss-Markov sources that are given in (6.70) and (6.71), respectively. For the function $g(R)$ we used the experimentally obtained approximation (5.59) for Gaussian pdfs. It should be noted that the corresponding functional relationships $\sigma_U^2(R)$ and $D(R)$ are only a rough approximation, since the distribution of the prediction residual U_n cannot be assumed to be Gaussian, at least not at low and medium rates.

In Fig. 6.8, the experimentally obtained data for DPCM coding with entropy-constrained scalar quantization and for entropy-constrained scalar quantization without prediction are compared to the derived operational distortion rate functions using the approximation $g(R)$ for Gaussian sources given in (5.59) and the information rate distortion function. For the shown experimental data and the derived operational distortion rate functions, the rate has been measured as the entropy of the quantizer output. The experimental data clearly indicate that DPCM coding significantly increases the rate distortion efficiency for sources with memory. Furthermore, we note that the derived operational distortion rate functions using the simple approximation for $g(R)$ represent suitable approximations for the experimentally obtained data. At high rates, the measured difference between the experimental

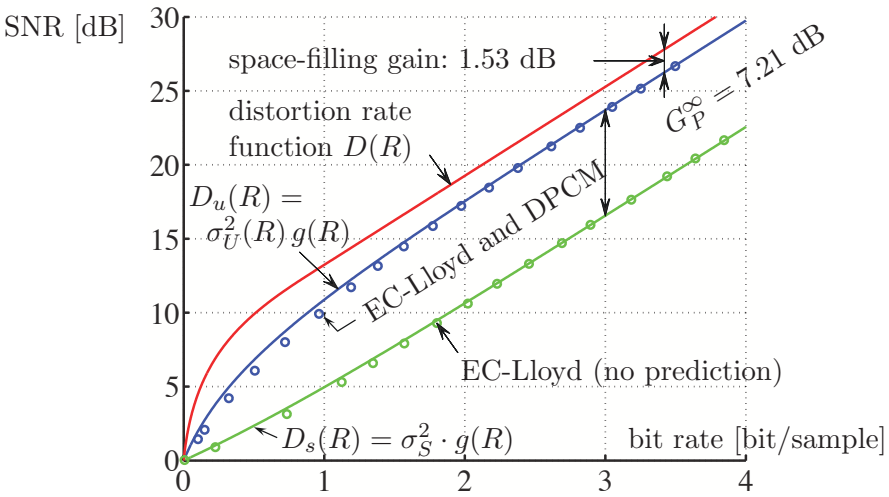


Fig. 6.8 Linear predictive coding of a stationary Gauss-Markov source with unit variance and a correlation factor of $\rho = 0.9$. The diagram compares the distortion rate efficiency of ECSQ (without prediction) and ECSQ inside the prediction loop to the (information) distortion rate function $D(R)$. The circles represent experimental data while the solid lines represent derived distortion rate functions. The rate is measured as the entropy of the quantizer output.

data for DPCM and the distortion rate bound is close to 1.53 dB, which corresponds to the space-filling gain of vector quantization as the quantizer dimension approaches infinity. This indicates that DPCM coding of stationary Gauss-Markov sources can fully exploit the dependencies inside the source at high rates and that the derived asymptotic operational distortion rate function (6.73) represents a reasonable approximation for distortion rate efficiency that can be obtained with DPCM coding of stationary Gauss-Markov sources at high rates. At low rates, the distance between the distortion rate bound and the obtained results for DPCM coding increases. A reason is that the variance σ_U^2 of the prediction residuals increases when the rate R is decreased, which is illustrated in Fig. 6.9.

The DPCM gain can be defined as the ratio of the operational distortion rate functions for scalar quantization and DPCM coding,

$$G_{\mathrm{DPCM}}(R) = \frac{\sigma_S^2 \cdot g_S(R)}{\sigma_U^2 \cdot g_U(R)}, \tag{6.74}$$

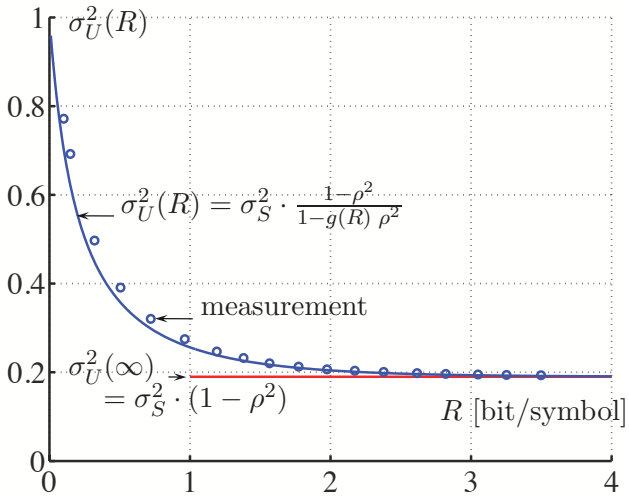


Fig. 6.9 Variance of prediction residual σ_U^2 as function of the bit rate for DPCM coding of a Gauss-Markov source with unit variance and a correlation factor of $\rho = 0.9$. The circles show the experimental results while the solid line represents the derived approximation. The rate is measured as the entropy of the quantizer output.

where $g_S(R)$ and $g_U(R)$ represent the normalized operational distortion rate functions for scalar quantization of the source signal and the prediction residuals, respectively. At high rates and under our intuitive assumption that the prediction residuals are nearly Gaussian, the normalized operational distortion rate function $g_U(R)$ for scalar quantization of the prediction residuals becomes equal to the normalized operational distortion rate function $g_S(R)$ for scalar quantization of the original samples. Then, the asymptotic coding gain for DPCM coding of stationary Gauss-Markov sources at high rates is approximately

$$G_{\mathrm{DPCM}}^{\infty}(R) = \frac{\sigma_S^2}{\sigma_U^2} = \frac{1}{1-\rho^2} = \frac{\frac{1}{2\pi}\int_{-\pi}^{\pi}\Phi_{SS}(\omega)d\omega}{2^{\frac{1}{2\pi}\int_{-\pi}^{\pi}\log_2\Phi_{SS}(\omega)d\omega}}. \tag{6.75}$$

6.4.2 Adaptive Differential Pulse Code Modulation

So far we have discussed linear prediction and DPCM coding for stationary sources. However, the input signals in practical coding systems are usually not stationary and thus a fixed predictor is not well suited. For nonstationary signals the predictor needs to be adapted based on

local signal characteristics. The adaptation method is either signaled from the sender to the receiver (forward adaptation) by side information or simultaneously derived at both sides using a prescribed algorithm (backward adaptation).

Forward Adaptive DPCM. A block diagram for a predictive codec with forward adaptation is shown in Fig. 6.10. The encoder sends new prediction coefficients to the decoder, which produces additional bit rate. It is important to balance the increased bit rate for the adaptation signal against the bit rate reduction resulting from improved prediction. In practical codecs, the adaptation signal is send infrequently at well-defined intervals. A typical choice in image and video coding is to adapt the predictor on a block-by-block basis.

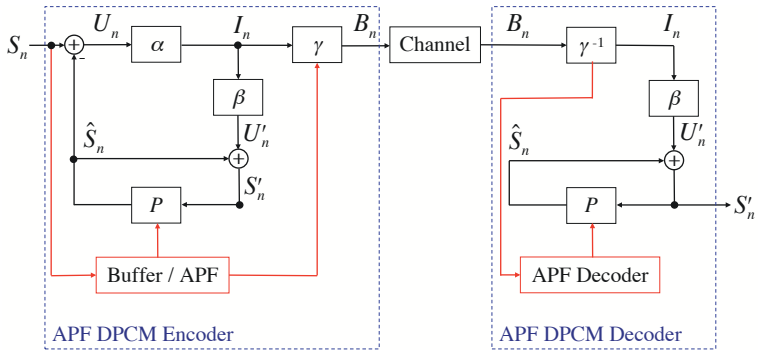


Fig. 6.10 Block diagram of a forward adaptive predictive codec.

Backward Adaptive DPCM. A block diagram for a predictive codec with backward adaptation is shown in Fig. 6.11. The prediction signal is derived from the previously decoded signal. It is advantageous relative to forward adaptation in that no additional bit rate is needed to signal the modifications of the predictor. Furthermore, backward adaptation does not introduced any additional encoding-decoding delay. The accuracy of the predictor is governed by the statistical properties of the source signal and the used adaptation algorithm. A drawback of backward adaptation is that the simultaneous computation of the adaptation signal is its increased sensitivity to transmission errors.

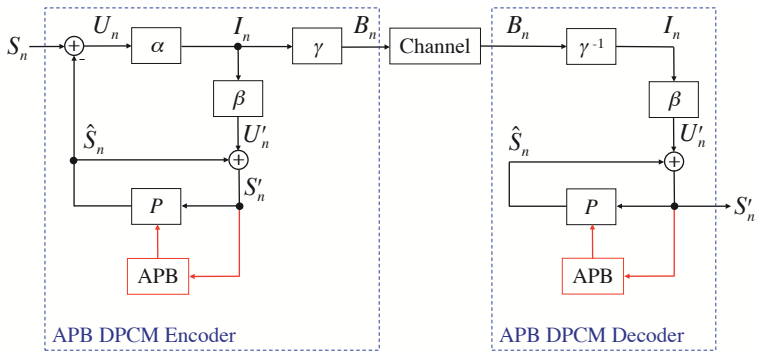


Fig. 6.11 Block diagram of a backward adaptive predictive codec.

6.5 Summary of Predictive Coding

In this chapter, we have discussed predictive coding. We introduced the concept of prediction as a procedure of estimating the value of a random variable based on already observed random variables. If the efficiency of a predictor is measured by the mean squared prediction error, the optimal prediction value is given by the conditional expectation of the random variable to be predicted given the observed random variables. For particular important sources such as Gaussian sources and autoregressive (AR) processes, the optimal predictor represents an affine function of the observation vector. A method to generally reduce the complexity of prediction is to constrain its structure to linear or affine prediction. The difference between linear and affine prediction is that the additional constant offset in affine prediction can compensate for the mean of the input signal.

For stationary random processes, the optimal linear predictor is given by the solution of the Yule-Walker equations and depends only on the autocovariances of the source signal. If an optimal affine predictor is used, the resulting prediction residual is orthogonal to each of the observed random variables. The optimal linear predictor for a stationary AR(m) process has m prediction coefficients, which are equal to the model parameters of the input process. A stationary Gauss-

Markov process is a stationary AR(1) process and hence the optimal linear predictor has a single prediction coefficient, which is equal to the correlation coefficient of the Gauss-Markov process. It is important to note that a non-matched predictor can increase the prediction error variance relative to the signal variance.

Differential pulse code modulation (DPCM) is the dominant structure for the combination of prediction and scalar quantization. In DPCM, the prediction is based on quantized samples. The combination of DPCM and entropy-constrained scalar quantization (ECSQ) has been analyzed in great detail for the special case of stationary Gauss-Markov processes. It has been shown that the prediction error variance is dependent on the bit rate. The derived approximation for high rates, which has been verified by experimental data, indicated that for stationary Gauss-Markov sources the combination of DPCM and ECSQ achieves the shape and memory gain of vector quantization at high rates.

7

Transform Coding

Similar to predictive coding, which we reviewed in the last chapter, transform coding is a concept for exploiting statistically dependencies of a source at a low complexity level. Transform coding is used in virtually all lossy image and video coding applications.

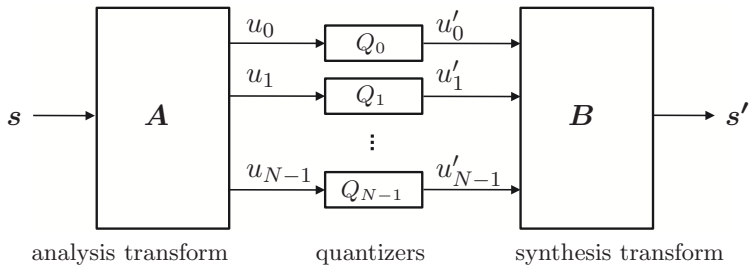


Fig. 7.1 Basic transform coding structure.

The basic structure of a typical transform coding system is shown in Fig. 7.1. A vector of a fixed number N input samples $\boldsymbol{s}$ is converted into a vector of N transform coefficients $\boldsymbol{u}$ using an analysis transform $\boldsymbol{A}$. The transform coefficients u_i, with $0 \leq i < N$, are quantized independently of each other using a set of scalar quantizers. The vector

of N reconstructed samples $\boldsymbol{s'}$ is obtained by transforming the vector of reconstructed transform coefficients $\boldsymbol{u'}$ using a synthesis transform $\boldsymbol{B}$.

In all practically used video coding systems, the analysis and synthesis transforms $\boldsymbol{A}$ and $\boldsymbol{B}$ are orthogonal block transforms. The sequence of source samples $\{s_n\}$ is partitioned into vectors $\boldsymbol{s}$ of adjacent samples and the transform coding consisting of an orthogonal analysis transform, scalar quantization of the transform coefficients, and an orthogonal synthesis transform is independently applied to each vector of samples. Since finally a vector $\boldsymbol{s}$ of sources samples is mapped to a vector $\boldsymbol{s'}$ of reconstructed samples, transform coding systems form a particular class of vector quantizers. The benefit in comparison to unconstrained vector quantization is that the imposed structural constraint allows implementations at a significantly lower complexity level.

The typical motivation for transform coding is the decorrelation and energy concentration effect. Transforms are designed in a way that, for typical input signals, the transform coefficients are much less correlated than the original source samples and the signal energy is concentrated in a few transform coefficients. As a result, the obtained transform coefficients have a different importance and simple scalar quantization becomes more effective in the transform domain than in the original signal space. Due to this effect, the memory advantage of vector quantization can be exploited to a large extend for typical source signals. Furthermore, by using entropy-constrained quantization for the transform coefficients also the shape advantage can be obtained. In comparison to unconstrained vector quantization, the rate distortion efficiency is basically reduced by the space-filling advantage, which can only be obtained by a significant increase in complexity.

For image and video coding applications, another advantage of transform coding is that the quantization in the transform domain often leads to an improvement of the subjective quality relative to a direct quantization of the source samples with the same distortion, in particular for low rates. The reason is that the transform coefficients contain information with different importance for the viewer and can therefore be treated differently. All perceptual distortion measures that are known to provide reasonable results weight the distortion in the trans-

form domain. The quantization of the transform coefficients can also be designed in a way that perceptual criteria are taken into account.

In contrast to video coding, the transforms that are used in still image coding are not restricted to the class of orthogonal block transforms. Instead, transforms that do not process the input signal on a block-by-block basis have been extensively studied and included into recent image coding standards. One of these transforms is the so-called discrete wavelet transform, which decomposes an image into components that correspond to band-pass filtered and downsampled versions of the image. Discrete wavelet transforms can be efficiently implemented using cascaded filter banks. Transform coding that is based on a discrete wavelet transform is also referred to as subband coding and is for example used in the JPEG 2000 standard [41, 72]. Another class of transforms are the lapped block transforms, which are basically applied on a block-by-block basis, but are characterized by basis functions that overlap the block boundaries. As a result, the transform coefficients for a block do not only depend on the samples inside the block, but also on samples of neighboring blocks. The vector of reconstructed samples for a block is obtained by transforming a vector that includes the transform coefficients of the block and of neighboring blocks. A hierarchical lapped transform with biorthogonal basis functions is included in the latest image coding standard JPEG XR [42]. The typical motivation for using wavelet transforms or lapped block transforms in image coding is that the nature of these transforms avoids the blocking artifacts which are obtained by transform coding with block-based transforms at low bit rates and are considered as one of the most disturbing coding artifacts. In video coding, wavelet transforms and lapped block transforms are rarely used due to the difficulties in efficiently combining these transforms with inter-picture prediction techniques.

In this chapter, we discuss transform coding with orthogonal block transforms, since this is the predominant transform coding structure in video coding. For further information on transform coding in general, the reader is referred to the tutorials [20] and [10]. An introduction to wavelet transforms and subband coding is given in the tutorials [74, 76] and [77]. As a reference for lapped blocks transforms and their application in image coding we recommend [62] and [53].

7.1 Structure of Transform Coding Systems

The basic structure of transform coding systems with block transforms is shown in Fig. 7.1. If we split the scalar quantizers Q_k, with $k = 0, \ldots, N-1$, into an encoder mapping α_k that converts the transform coefficients into quantization indexes and a decoder mapping β_k that converts the quantization indexes into reconstructed transform coefficients and additionally introduce a lossless coding γ for the quantization indexes, we can decompose the transform coding system shown in Fig. 7.1 into a transform encoder and a transform decoder as illustrated in Fig. 7.2.

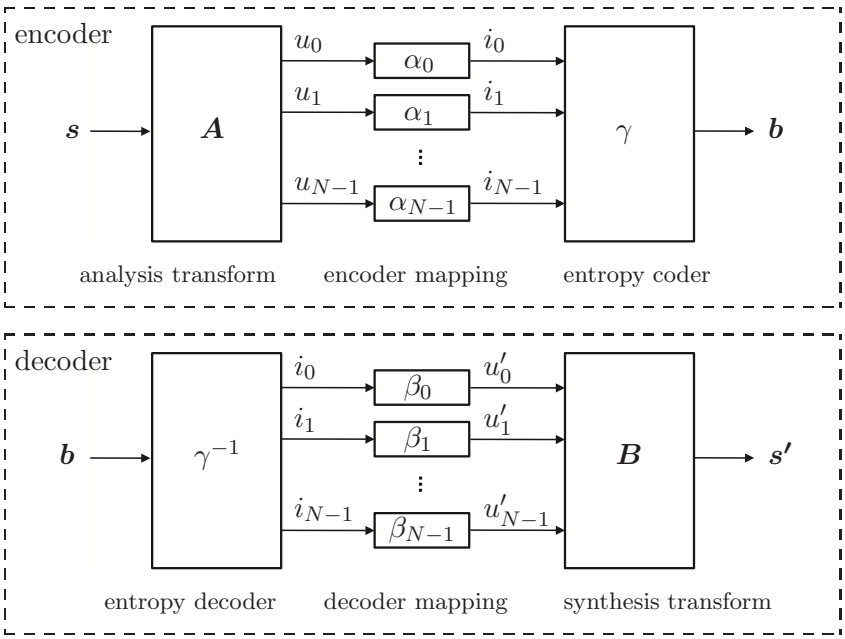


Fig. 7.2 Encoder and decoder of a transform coding system.

In the transform encoder, the analysis transform converts a vector $\boldsymbol{s} = (s_0, \cdots, s_{N-1})^T$ of N source samples into a vector of N transform coefficients $\boldsymbol{u} = (u_0, \cdots, u_{N-1})^T$. Each transform coefficient u_k is then mapped onto a quantization index i_k using an encoder mapping α_k. The quantization indexes of all transform coefficients are coded using a lossless mapping γ, resulting in a sequence of codewords $\boldsymbol{b}$.

In the transform decoder, the sequence of codewords $\boldsymbol{b}$ is mapped to the set of quantization indexes i_k using the inverse lossless mapping γ^{-1}. The decoder mappings β_k convert the quantization indexes i_k into reconstructed transform coefficients u'_k. The vector of N reconstructed samples $\boldsymbol{s'} = (s'_0, \cdots, s'_{N-1})^T$ is obtained by transforming the vector of N reconstructed transform coefficients $\boldsymbol{u'} = (u'_0, \cdots, u'_{N-1})^T$ using the synthesis transform.

7.2 Orthogonal Block Transforms

In the following discussion of transform coding, we restrict our considerations to stationary sources and transform coding systems with the following properties:

(1) *Linear block transforms*: The analysis and synthesis transform are linear block transforms.

(2) *Perfect reconstruction*: The synthesis transform is the inverse of the analysis transform.

(3) *Orthonormal basis*: The basis vectors of the analysis transform form an orthonormal basis.

Linear Block Transforms. For linear block transforms of size N, each component of an N-dimensional output vector represents a linear combination of the components of the N-dimensional input vector. A linear block transform can be written as a matrix multiplication. The analysis transform, which maps a vector of source samples $\boldsymbol{s}$ to a vector of transform coefficients $\boldsymbol{u}$, is given by

$$\boldsymbol{u} = \boldsymbol{A}\ \boldsymbol{s}, \tag{7.1}$$

where the matrix $\boldsymbol{A}$ is referred to as the analysis transform matrix. Similarly, the synthesis transform, which maps a vector of reconstructed transform coefficients $\boldsymbol{u'}$ to a vector of reconstructed samples $\boldsymbol{s'}$, can be written as

$$\boldsymbol{s'} = \boldsymbol{B}\ \boldsymbol{u'}, \tag{7.2}$$

where the matrix $\boldsymbol{B}$ represents the synthesis transform matrix.

Perfect Reconstruction. The perfect reconstruction property specifies that the synthesis transform matrix is the inverse of the analysis transform matrix, $\boldsymbol{B} = \boldsymbol{A}^{-1}$. If the transform coefficients are not quantized, i.e., if $\boldsymbol{u'} = \boldsymbol{u}$, the vector of reconstructed samples is equal to the vector of source samples,

$$\boldsymbol{s'} = \boldsymbol{B}\ \boldsymbol{u} = \boldsymbol{B}\ \boldsymbol{A}\ \boldsymbol{s} = \boldsymbol{A}^{-1}\ \boldsymbol{A}\ \boldsymbol{s} = \boldsymbol{s}. \tag{7.3}$$

If an invertible analysis transform $\boldsymbol{A}$ produces independent transform coefficients and the component quantizers reconstruct the centroids of the quantization intervals, the inverse of the analysis transform is the optimal synthesis transform in the sense that it yields the minimum distortion among all linear transforms given the coded transform coefficients. It should, however, be noted that if these conditions are not fulfilled, a synthesis transform $\boldsymbol{B}$ that is not equal to the inverse of the analysis transform may reduce the distortion [20].

Orthonormal Basis. An analysis transform matrix $\boldsymbol{A}$ forms an orthonormal basis if its basis vectors given by the rows of the matrix are orthogonal to each other and have the length 1. Matrices with this property are referred to as *unitary matrices*. The corresponding transform is said to be an *orthogonal transform*. The inverse of a unitary matrix $\boldsymbol{A}$ is its conjugate transpose, $\boldsymbol{A}^{-1} = \boldsymbol{A}^{\dagger}$. A unitary matrix with real entries is called an *orthogonal matrix* and its inverse is equal to its transpose, $\boldsymbol{A}^{-1} = \boldsymbol{A}^{T}$. For linear transform coding systems with the perfect reconstruction property and orthogonal matrices, the synthesis transform is given by

$$\boldsymbol{s'} = \boldsymbol{B}\ \boldsymbol{u'} = \boldsymbol{A}^{T}\ \boldsymbol{u'}. \tag{7.4}$$

Unitary transform matrices are often desirable, because the mean square error between a reconstruction and source vector can be minimized with independent scalar quantization of the transform coefficients. Furthermore, as we will show below, the distortion in the transform domain is equal to the distortion in the original signal space. In practical transform coding systems, it is usually sufficient to require that the basis vectors are orthogonal to each other. The different norms can be easily taken into account in the quantizer design.

We can consider a linear analysis transform $\boldsymbol{A}$ as optimal if the transform coding system consisting of the analysis transform $\boldsymbol{A}$, optimal entropy-constrained scalar quantizers for the transform coefficients (which depend on the analysis transform), and the synthesis transform $\boldsymbol{B} = \boldsymbol{A}^{-1}$ yields a distortion for a particular given rate that is not greater than the distortion that would be obtained with any other transform at the same rate. In this respect, a unitary transform is optimal for the MSE distortion measure if it produces independent transform coefficients. Such a transform does, however, not exist for all sources. Depending on the source signal, a non-unitary transform may be superior [20, 13].

Properties of Orthogonal Block Transforms. An important property of transform coding systems with the perfect reconstruction property and unitary transforms is that the MSE distortion is preserved in the transform domain. For the general case of complex transform matrices, the MSE distortion between the reconstructed samples and the source samples can be written as

$$\begin{aligned} d_N(\boldsymbol{s}, \boldsymbol{s'}) &= \frac{1}{N}\,(\boldsymbol{s} - \boldsymbol{s'})^\dagger\,(\boldsymbol{s} - \boldsymbol{s'}) \\ &= \frac{1}{N}\,\big(\boldsymbol{A}^{-1}\,\boldsymbol{u} - \boldsymbol{B}\,\boldsymbol{u'}\big)^\dagger\,\big(\boldsymbol{A}^{-1}\,\boldsymbol{u} - \boldsymbol{B}\,\boldsymbol{u'}\big), \end{aligned} \tag{7.5}$$

where $\dagger$ denotes the conjugate transpose. With the properties of perfect reconstruction and unitary transforms ($\boldsymbol{B} = \boldsymbol{A}^{-1} = \boldsymbol{A}^\dagger$), we obtain

$$\begin{aligned} d_N(\boldsymbol{s}, \boldsymbol{s'}) &= \frac{1}{N}\,\big(\boldsymbol{A}^\dagger\,\boldsymbol{u} - \boldsymbol{A}^\dagger\,\boldsymbol{u'}\big)^\dagger\,\big(\boldsymbol{A}^\dagger\,\boldsymbol{u} - \boldsymbol{A}^\dagger\,\boldsymbol{u'}\big) \\ &= \frac{1}{N}\,(\boldsymbol{u} - \boldsymbol{u'})^\dagger\,\boldsymbol{A}\,\boldsymbol{A}^{-1}\,(\boldsymbol{u} - \boldsymbol{u'}) \\ &= \frac{1}{N}\,(\boldsymbol{u} - \boldsymbol{u'})^\dagger\,(\boldsymbol{u} - \boldsymbol{u'}) = d_N(\boldsymbol{u}, \boldsymbol{u'}). \end{aligned} \tag{7.6}$$

For the special case of orthogonal transform matrices, the conjugate transposes in the above derivation can be replaced with the transposes, which yields the same result. Scalar quantization that minimizes the MSE distortion in the transform domain also minimizes the MSE distortion in the original signal space.

Another important property for orthogonal transforms can be derived by considering the autocovariance matrix for the random vectors $\boldsymbol{U}$ of transform coefficients,

$$\boldsymbol{C}_{UU} = E\big\{(\boldsymbol{U} - E\{\boldsymbol{U}\})(\boldsymbol{U} - E\{\boldsymbol{U}\})^T\big\}\,. \tag{7.7}$$

With $\boldsymbol{U} = \boldsymbol{A}\,\boldsymbol{S}$ and $\boldsymbol{A}^{-1} = \boldsymbol{A}^T$, we obtain

$$\boldsymbol{C}_{UU} = E\big\{\boldsymbol{A}\,(\boldsymbol{S} - E\{\boldsymbol{S}\})(\boldsymbol{S} - E\{\boldsymbol{S}\})^T\,\boldsymbol{A}^T\big\} = \boldsymbol{A}\,\boldsymbol{C}_{SS}\,\boldsymbol{A}^{-1}, \tag{7.8}$$

where $\boldsymbol{C}_{SS}$ denotes the autocovariance matrix for the random vectors $\boldsymbol{S}$ of original source samples. It is known from linear algebra that the trace $\mathrm{tr}(\boldsymbol{X})$ of a matrix $\boldsymbol{X}$ is similarity-invariant,

$$\mathrm{tr}(\boldsymbol{X}) = \mathrm{tr}(\boldsymbol{P}\,\boldsymbol{X}\,\boldsymbol{P}^{-1}), \tag{7.9}$$

with $\boldsymbol{P}$ being an arbitrary invertible matrix. Since the trace of an autocovariance matrix is the sum of the variances of the vector components, the arithmetic mean of the variances σ_i^2 of the transform coefficients is equal to the variance σ_S^2 of the original samples,

$$\frac{1}{N}\sum_{i=0}^{N-1}\sigma_i^2 = \sigma_S^2. \tag{7.10}$$

Geometrical Interpretation. An interpretation of the matrix multiplication in (7.2) is that the vector of reconstructed samples $\boldsymbol{s'}$ is represented as a linear combination of the columns of the synthesis transform matrix $\boldsymbol{B}$, which are also referred to as the basis vectors $\boldsymbol{b}_k$ of the synthesis transform. The weights in this linear combination are given by the reconstructed transform coefficients u'_k and we can write

$$\boldsymbol{s'} = \sum_{k=0}^{N-1} u'_k\,\boldsymbol{b}_k = u'_0\,\boldsymbol{b}_0 + u'_1\,\boldsymbol{b}_1 + \cdots + u'_{N-1}\,\boldsymbol{b}_{N-1}. \tag{7.11}$$

Similarly, the original signal vector $\boldsymbol{s}$ is represented by a linear combination of the basis vectors $\boldsymbol{a}_k$ of the inverse analysis transform, given by the columns of $\boldsymbol{A}^{-1}$,

$$\boldsymbol{s} = \sum_{k=0}^{N-1} u_k\,\boldsymbol{a}_k = u_0\,\boldsymbol{a}_0 + u_1\,\boldsymbol{a}_1 + \cdots + u_{N-1}\,\boldsymbol{a}_{N-1}, \tag{7.12}$$

where the weighting factors are the transform coefficients u_k. If the analysis transform matrix is orthogonal ($\boldsymbol{A}^{-1} = \boldsymbol{A}^T$), the columns of $\boldsymbol{A}^{-1}$ are equal to the rows of $\boldsymbol{A}$. Furthermore, the basis vectors $\boldsymbol{a}_k$ are orthogonal to each other and build a coordinate system with perpendicular axes. Hence, there is a unique way to represent a signal vector $\boldsymbol{s}$ in the new coordinate system given by the set of basis vectors $\{\boldsymbol{a}_k\}$. Each transform coefficient u_k is given by the projection of the signal vector $\boldsymbol{s}$ onto the corresponding basis vector $\boldsymbol{a}_k$, which can be written as scalar product

$$u_k = \boldsymbol{a}_k^T \, \boldsymbol{s}. \tag{7.13}$$

Since the coordinate system spanned by the basis vectors has perpendicular axes and the origin coincides with the origin of the signal coordinate system, an orthogonal transform specifies rotations and reflections in the N-dimensional Euclidean space. If the perfect reconstruction property ($\boldsymbol{B} = \boldsymbol{A}^{-1}$) is fulfilled, the basis vectors $\boldsymbol{b}_k$ of the synthesis transform are equal to the basis vectors $\boldsymbol{a}_k$ of the analysis transform and the synthesis transform specifies the inverse rotations and reflections of the analysis transform.

As a simple example, we consider the following orthogonal 2×2 synthesis matrix,

$$\boldsymbol{B} = \begin{bmatrix} \boldsymbol{b}_0 & \boldsymbol{b}_1 \end{bmatrix} = \frac{1}{\sqrt{2}} \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix}. \tag{7.14}$$

The analysis transform matrix $\boldsymbol{A}$ is given by the transpose of the synthesis matrix, $\boldsymbol{A} = \boldsymbol{B}^T$. The transform coefficients u_k for a given signal vector $\boldsymbol{s}$ are the scalar products of the signal vector $\boldsymbol{s}$ and the basis vectors $\boldsymbol{b}_k$. For a signal vector $\boldsymbol{s}$=$[4, 3]^T$, we obtain

$$u_0 = \boldsymbol{b}_0^T \cdot \boldsymbol{s} = (4+3)/\sqrt{2} = 3.5 \cdot \sqrt{2}, \tag{7.15}$$

$$u_1 = \boldsymbol{b}_1^T \cdot \boldsymbol{s} = (4-3)/\sqrt{2} = 0.5 \cdot \sqrt{2}. \tag{7.16}$$

The signal vector $\boldsymbol{s}$ is represented as a linear combination of the basis vectors, where the weights are given by the transform coefficients,

$$\begin{aligned} \boldsymbol{s} &= u_0 \cdot \boldsymbol{b}_0 + u_1 \cdot \boldsymbol{b}_1 \\ \begin{bmatrix} 4 \\ 3 \end{bmatrix} &= (3.5 \cdot \sqrt{2}) \cdot \frac{1}{\sqrt{2}} \begin{bmatrix} 1 \\ 1 \end{bmatrix} (0.5 \cdot \sqrt{2}) \cdot \frac{1}{\sqrt{2}} \begin{bmatrix} 1 \\ -1 \end{bmatrix}. \end{aligned} \tag{7.17}$$

As illustrated in Fig. 7.3, the coordinate system spanned by the basis vectors $\boldsymbol{b}_0$ and $\boldsymbol{b}_1$ is rotated by 45 degrees relative to the original coordinate system. The transform coefficients specify the projections of the signal vector $\boldsymbol{s}$ onto the axes of the new coordinate system.

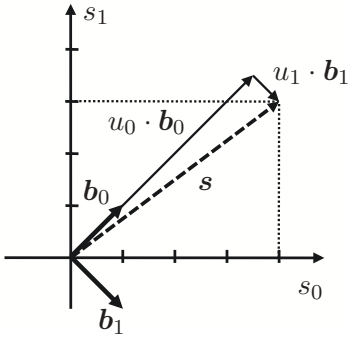


Fig. 7.3 Geometric interpretation of an orthogonal 2×2 transform.

Fig. 7.4 illustrates the effect of a decorrelating orthogonal transform on the example of the given 2×2 transform for stationary zero-mean Gauss-Markov sources with unit variance and different correlation coefficients ρ. If the source samples are not correlated ($\rho = 0$), the transform does not have any effect. But for correlated sources, the transform rotates the distribution of the source vectors in a way that the primary axes of the distribution are aligned with axes of the coordinate system in the transform domain. For the example 2×2 transform this has the effect that the variance for one transform coefficient is minimized while the variance of the other transform coefficient is maximized. The signal energy is shifted toward the first transform coefficient U_0.

In Fig. 7.5 the quantization cells for scalar quantization in the original signal space are compared with the quantization cells for transform coding. As discussed in Chapter 5, the effective quantization cells for simple scalar quantization in the N-dimensional signal space are hyperrectangles that are aligned with the axes of the coordinate system as illustrated in the left diagram of Fig. 7.5. For transform coding, the quantization cells in the transform domain are hyperrectangles that are aligned with the axes of the coordinate system of the transform coefficients (middle diagram of Fig. 7.5). In the original signal space,

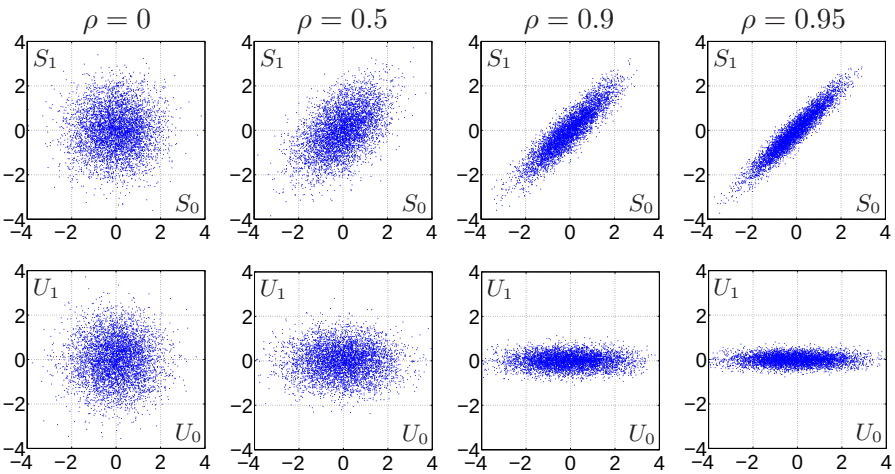


Fig. 7.4 Effect of a decorrelating orthogonal transform on the example of the 2×2 transform given in (7.14) for stationary Gauss-Markov sources with zero mean, unit variance and different correlation coefficients ρ: (top) distribution of sources vectors; (bottom) distribution of transform coefficient vectors.

the quantization cells are still hyperrectangles, but the grid of quantization cells is rotated and aligned with the basis vectors of the orthogonal transform as shown in the right diagram of Fig. 7.5. As a rough approximation, the required bit rate can be considered as proportional to the number of quantization cells associated with appreciable probabilities in the coordinate directions of the quantization grid. This indicates that, for correlated sources, transform coding yields a higher rate distortion efficiency than scalar quantization in the original domain.

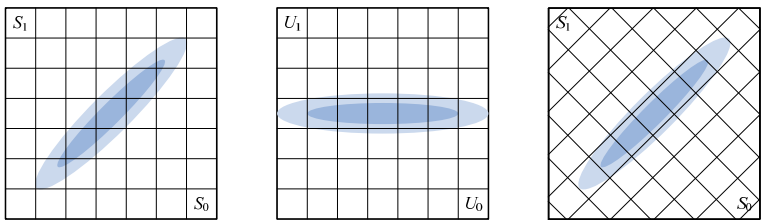


Fig. 7.5 Comparison of transform coding and scalar quantization in the original signal space: (left) source distribution and quantization cells for scalar quantization; (middle) distribution of transform coefficients and quantization cells in the transform domain; (right) source distribution and quantization cells for transform coding in the original signal space.

7.3 Bit Allocation for Transform Coefficients

Before we discuss decorrelating transforms in more detail, we analyze the problem of bit allocation for transform coefficients. As mentioned above, the transform coefficients have usually a different importance and hence the overall rate distortion efficiency of a transform coding system depends on a suitable distribution of the overall rate R among the transform coefficients. A bit allocation is optimal if a given overall rate R is distributed in a way that the resulting overall distortion D is minimized. If we use the MSE distortion measure, the distortion in the original signal space is equal to the distortion in the transform domain. Hence, with R_i representing the component rates for the transform coefficients u_i and $D_i(R_i)$ being the operational distortion rate functions for the component quantizers, we want to minimize

$$D(R) = \frac{1}{N}\sum_{i=0}^{N-1} D_i(R_i) \qquad \text{subject to} \qquad \frac{1}{N}\sum_{i=0}^{N-1} R_i = R. \tag{7.18}$$

As has been discussed in sec. 5.2.2, the constrained optimization problem (7.18) can be reformulated as an unconstrained minimization of the Lagrangian cost functional $J = D + \lambda R$. If we assume that the operational distortion rate functions $D_i(R_i)$ for the component quantizers are convex, the optimal rate allocation can be found by setting the partial derivatives of the Lagrangian functional J with respect to the component rates R_i equal to 0,

$$\frac{\mathrm{d}}{\mathrm{d}R_i}\left(\frac{1}{N}\sum_{i=1}^{N} D_i(R_i) + \frac{\lambda}{N}\sum_{i=1}^{N} R_i\right) = \frac{1}{N}\frac{\mathrm{d}D_i(R_i)}{\mathrm{d}R_i} + \frac{\lambda}{N} = 0, \tag{7.19}$$

which yields

$$\frac{\mathrm{d}}{\mathrm{d}R_i} D_i(R_i) = -\lambda = \text{const.} \tag{7.20}$$

This so-called Pareto condition states that, for optimal bit allocation, all component quantizers should be operated at equal slopes of their operational distortion rate functions $D_i(R_i)$.

In sec. 5.2.4, we have shown that the operational distortion rate function of scalar quantizers can be written as

$$D_i(R_i) = \sigma_i^2 \cdot g_i(R_i), \tag{7.21}$$

where σ_i^2 is the variance of the input source and $g_i(R_i)$ is the operational distortion rate function for the normalized distribution with unit variance. In general, it is justified to assume that $g_i(R_i)$ is a nonnegative, strictly convex function and has a continuous first derivative $g_i'(R_i)$ with $g_i'(\infty) = 0$. Then, the Pareto condition yields

$$-\sigma_i^2 \; g_i'(R_i) = \lambda. \tag{7.22}$$

As discussed in sec. 4.4, it has to be taken into account that the component rate R_i for a particular transform coefficient cannot be negative. If $\lambda \geq -\sigma_i^2 \; g_i'(0)$, the quantizer for the transform coefficient u_i cannot be operated at the given slope λ. In this case, it is optimal to set the component rate R_i equal to zero. The overall distortion is minimized if the overall rate is spent for coding only the transform coefficients with $-\sigma_i^2 \; g_i'(0) > \lambda$. This yields the following bit allocation rule,

$$R_i = \begin{cases} 0 & : \; -\sigma_i^2 g_i'(0) \leq \lambda \\ \eta_i\left(-\frac{\lambda}{\sigma_i^2}\right) & : \; -\sigma_i^2 g_i'(0) > \lambda \end{cases}, \tag{7.23}$$

where $\eta_i(\cdot)$ denotes the inverse of the derivative $g_i'(\cdot)$. Since $g_i'(R_i)$ is a continuous strictly increasing function for $R_i \geq 0$ with $g_i'(\infty) = 0$, the inverse $\eta_i(x)$ is a continuous strictly increasing function for the range $g_i'(0) \leq x \leq 0$ with $\eta_i(f_i'(0)) = 0$ and $\eta_i(0) = \infty$.

7.3.1 Approximation for Gaussian Sources

If the input signal has a Gaussian distribution, the distributions for all transform coefficients are also Gaussian, since the signal for each transform coefficient represents a linear combination of Gaussian sources. Hence, we can assume that the operational distortion rate function for all component quantizers is given by

$$D_i(R_i) = \sigma_i^2 \cdot g(R), \tag{7.24}$$

where $g(R)$ represents the operational distortion rate function for Gaussian sources with unit variance. In order to derive an approximate formula for the optimal bit allocation, we assume that the component quantizers are entropy-constrained scalar quantizers and use the approximation (5.59) for $g(R)$ that has been experimentally

found for entropy-constrained scalar quantization of Gaussian sources in sec. 5.2.4,

$$g(R) = \frac{\varepsilon^2}{a} \ln(a \cdot 2^{-2R} + 1). \tag{7.25}$$

The factor ε^2 is equal to $\pi e/6$ and the model parameter a is approximately 0.9519. The derivative $g'(R)$ and its inverse $\eta(x)$ are given by

$$g'(R) = -\frac{\varepsilon^2 \cdot 2\ln 2}{a + 2^{2R}}, \tag{7.26}$$

$$\eta(x) = \frac{1}{2}\log_2\left(-\frac{\varepsilon^2 \cdot 2\ln 2}{x} - a\right). \tag{7.27}$$

As stated above, for an optimal bit allocation, the component rate R_i for a transform coefficient has to be set equal to 0, if

$$\lambda \geq -\sigma_i^2 g'(0) = \sigma_i^2 \frac{\varepsilon^2 \cdot 2\ln 2}{a+1}. \tag{7.28}$$

With the parameter

$$\theta = \lambda \frac{a+1}{\varepsilon^2 \cdot 2\ln 2}, \tag{7.29}$$

we obtain the bit allocation rule

$$R_i(\theta) = \begin{cases} 0 & : \quad \theta \geq \sigma_i^2 \\ \frac{1}{2}\log_2\left(\frac{\sigma_i^2}{\theta}(a+1) - a\right) & : \quad \theta < \sigma_i^2 \end{cases}. \tag{7.30}$$

The resulting component distortions are given by

$$D_i(\theta) = \begin{cases} \sigma_i^2 & : \quad \theta \geq \sigma_i^2 \\ -\frac{\varepsilon^2 \ln 2}{a} \cdot \sigma_i^2 \cdot \log_2\left(1 - \frac{\theta}{\sigma_i^2}\frac{a}{a+1}\right) & : \quad \theta < \sigma_i^2 \end{cases}. \tag{7.31}$$

If the variances σ_i^2 of the transform coefficients are known, the approximation of the operational distortion rate function for transform coding of Gaussian sources with entropy-constrained scalar quantization is given by the parametric formulation

$$R(\theta) = \frac{1}{N}\sum_{i=0}^{N-1} R_i(\theta), \qquad D(\theta) = \frac{1}{N}\sum_{i=0}^{N-1} D_i(\theta), \tag{7.32}$$

where $R(\theta)$ and $D(\theta)$ are specified by (7.30) and (7.31), respectively. The approximation of the operational distortion rate function can be calculated by varying the parameter θ in the range from 0 to $\sigma_{\max}^2$, with $\sigma_{\max}^2$ being the maximum variance of the transform coefficients.

7.3.2 High-Rate Approximation

In the following, we assume that the overall rate R is high enough so that all component quantizers are operated at high component rates R_i. In sec. 5.2.3, we have shown that the asymptotic operational distortion rate functions for scalar quantizers can be written as

$$D_i(R_i) = \varepsilon_i^2 \, \sigma_i^2 \, 2^{-2R_i}, \tag{7.33}$$

where the factor ε_i^2 depends only on the type of the source distribution and the used scalar quantizer. Using these high rate approximations for the component quantizers, the Pareto condition becomes

$$\frac{\mathrm{d}}{\mathrm{d}R_i} D_i(R_i) = -2 \, \ln 2 \, \varepsilon_i^2 \, \sigma_i^{2\,-2R_i} = -2 \, \ln 2 \, D_i(R_i) = \text{const.} \tag{7.34}$$

At high rates, an optimal bit allocation is obtained if all component distortions D_i are the same. Setting the component distortions D_i equal to the overall distortion D, yields

$$R_i(D) = \frac{1}{2} \log_2 \left(\frac{\sigma_i^2 \varepsilon_i^2}{D} \right). \tag{7.35}$$

For the overall operational rate distortion function, we obtain

$$R(D) = \frac{1}{N} \sum_{i=0}^{N-1} R_i(D) = \frac{1}{2N} \sum_{i=0}^{N-1} \log_2 \left(\frac{\sigma_i^2 \varepsilon_i^2}{D} \right) \tag{7.36}$$

With the geometric means of the variances σ_i^2 and the factors ϵ_i^2,

$$\tilde{\sigma}^2 = \left(\prod_{i=0}^{N-1} \sigma_i^2 \right)^{\frac{1}{N}} \qquad \text{and} \qquad \tilde{\varepsilon}^2 = \left(\prod_{i=0}^{N-1} \varepsilon_i^2 \right)^{\frac{1}{N}}, \tag{7.37}$$

the asymptotic operational distortion rate function for high rates can be written as

$$D(R) = \tilde{\varepsilon}^2 \cdot \tilde{\sigma}^2 \cdot 2^{-2R}. \tag{7.38}$$

It should be noted that this result can also be derived without using the Pareto condition, which was obtained by calculus. Instead, we can use the inequality of arithmetic and geometric means and derive the high rate approximation similar to the rate distortion function for Gaussian sources with memory in sec. 4.4.

For Gaussian sources, all transform coefficients have a Gaussian distribution (see sec. 7.3.1), and thus all factors ε_i^2 are the same. If entropy-constrained scalar quantizers are used, the factors ε_i^2 are equal to $\pi e/6$ (see sec. 5.2.3) and the asymptotic operational distortion rate function for high rates is given by

$$D(R) = \frac{\pi e}{6} \cdot \tilde{\sigma}^2 \cdot 2^{-2R}. \tag{7.39}$$

Transform Coding Gain. The effectiveness of a transform is often specified by the transform coding gain, which is defined as the ratio of the operational distortion rate functions for scalar quantization and transform coding. At high rates, the transform coding gain is given by

$$G_T = \frac{\varepsilon_S^2 \cdot \sigma_S^2 \cdot 2^{-2R}}{\tilde{\varepsilon}^2 \cdot \tilde{\sigma}^2 \cdot 2^{-2R}}, \tag{7.40}$$

where ε_S^2 is the factor of the high rate approximation of the operational distortion rate function for scalar quantization in the original signal space and σ_S^2 is the variance of the input signal.

By using the relationship (7.10), the high rate transform gain for Gaussian sources can be expressed as the ratio of the arithmetic and geometric mean of the transform coefficient variances,

$$G_T = \frac{\frac{1}{N}\sum_{i=0}^{N-1} \sigma_i^2}{\sqrt[N]{\prod_{i=0}^{N-1} \sigma_i^2}}. \tag{7.41}$$

The high rate transform gain for Gaussian sources is maximized if the geometric mean is minimized. The transform that minimizes the geometric mean is the Karhunen Loève Transform, which will be discussed in the next section.

7.4 The Karhunen Loève Transform (KLT)

Due to its importance in the theoretical analysis of transform coding we discuss the Karhunen Loève Transform (KLT) in some detail in the following. The KLT is an orthogonal transform that decorrelates the vectors of input samples. The transform matrix $\boldsymbol{A}$ is dependent on the statistics of the input signal.

Let $\boldsymbol{S}$ represent the random vectors of original samples of a stationary input sources. The random vectors of transform coefficients are given by $\boldsymbol{U} = \boldsymbol{A}\,\boldsymbol{S}$ and for the autocorrelation matrix of the transform coefficients we obtain

$$\boldsymbol{R}_{UU} = E\{\boldsymbol{U}\,\boldsymbol{U}^T\} = E\{(\boldsymbol{A}\boldsymbol{S})(\boldsymbol{A}\boldsymbol{S})^T\} = \boldsymbol{A}\boldsymbol{R}_{SS}\boldsymbol{A}^T, \tag{7.42}$$

where

$$\boldsymbol{R}_{SS} = E\{\boldsymbol{S}\boldsymbol{S}^T\} \tag{7.43}$$

denotes the autocorrelation matrix of the input process. To get uncorrelated transform coefficients, the orthogonal transform matrix $\boldsymbol{A}$ has to be chosen in a way that the autocorrelation matrix $\boldsymbol{R}_{UU}$ becomes a diagonal matrix. Equation (7.42) can be slightly reformulated as

$$\boldsymbol{R}_{SS}\boldsymbol{A}^T = \boldsymbol{A}^T\boldsymbol{R}_{SS}. \tag{7.44}$$

With $\boldsymbol{b}_i$ representing the basis vectors of the synthesis transform, i.e., the column vectors of $\boldsymbol{A}^{-1} = \boldsymbol{A}^T$ and the row vectors of $\boldsymbol{A}$, it becomes obvious that $\boldsymbol{R}_{UU}$ is a diagonal matrix if the eigenvector equation

$$\boldsymbol{R}_{SS}\,\boldsymbol{b}_i = \xi_i\,\boldsymbol{b}_i \tag{7.45}$$

is fulfilled for all basis vectors $\boldsymbol{b}_i$. The eigenvalues ξ_i represents the elements r_{ii} on the main diagonal of the diagonal matrix $\boldsymbol{R}_{UU}$. The rows of the transform matrix $\boldsymbol{A}$ are build by a set of unit-norm eigenvectors of $\boldsymbol{R}_{SS}$ that are orthogonal to each other. The autocorrelation matrix for the transform coefficients $\boldsymbol{R}_{UU}$ is a diagonal matrix with the eigenvalues of $\boldsymbol{R}_{SS}$ on its main diagonal. The transform coefficient variances σ_i^2 are equal to the eigenvalues ξ_i of the autocorrelation matrix $\boldsymbol{R}_{SS}$.

A KLT exists for all sources, since symmetric matrices as the autocorrelation matrix $\boldsymbol{R}_{SS}$ are always orthogonally diagonizable. There exist more than one KLT of any particular size $N > 1$ for all stationary sources, because the rows of $\boldsymbol{A}$ can be multiplied by -1 or permuted without influencing the orthogonality of $\boldsymbol{A}$ or the diagonal form of $\boldsymbol{R}_{UU}$. If the eigenvalues of $\boldsymbol{R}_{SS}$ are not distinct, there are additional degrees of freedom for constructing KLT transform matrices. Numerical methods for calculating the eigendecomposition $\boldsymbol{R}_{SS} = \boldsymbol{A}^T\mathrm{diag}(\xi_i)\boldsymbol{A}$ of real symmetric matrices $\boldsymbol{R}_{SS}$ are the classical and the cyclic Jacobi algorithm [43, 18].

Nonstationary Sources. For nonstationary sources, transform coding with a single KLT transform matrix is suboptimal. Similar to the predictor in predictive coding, the transform matrix should be adapted based on the local signal statistics. The adaptation can be realized either as forward adaptation or as backward adaptation. With forward adaptive techniques, the transform matrix is estimated at the encoder and an adaptation signal is transmitted as side information, which increases the overall bit rate and usually introduces an additional delay. In backward adaptive schemes, the transform matrix is simultaneously estimated at the encoder and decoder side based on already coded samples. Forward adaptive transform coding is discussed in [12] and transform coding with backward adaptation is investigated in [21].

7.4.1 On the Optimality of the KLT

We showed that the KLT is an orthogonal transform that yields decorrelated transform coefficients. In the following, we show that the KLT is also the orthogonal transform that maximizes the rate distortion efficiency for stationary zero-mean Gaussian sources if optimal scalar quantizers are used for quantizing the transform coefficients. The following proof was first delineated in [19].

We consider a transform coding system with an orthogonal $N \times N$ analysis transform matrix $\boldsymbol{A}$, the synthesis transform matrix $\boldsymbol{B} = \boldsymbol{A}^T$, and scalar quantization of the transform coefficients. We further assume that we use a set of scalar quantizers that are given by scaled versions of a quantizer for unit variance for which the operational distortion rate function is given by a nonincreasing function $g(R)$. The decision thresholds and reconstruction levels of the quantizers are scaled according to the variances of the transform coefficients. Then, the operational distortion rate function for each component quantizer is given by

$$D_i(R_i) = \sigma_i^2 \cdot g(R_i), \tag{7.46}$$

where σ_i^2 denotes the variance of the corresponding transform coefficient (cp. sec. 5.2.4). It should be noted that such a setup is optimal for Gaussian sources if the function $g(R)$ is the operational distortion rate function of an optimal scalar quantizer. The optimality of a

quantizer may depend on the application. As an example, we could consider entropy-constrained Lloyd quantizers as optimal if we assume a lossless coding that achieves an average codeword length close to the entropy. For Gaussian sources, the transform coefficients have also a Gaussian distribution. The corresponding optimal component quantizers are scaled versions of the optimal quantizer for unit variance and their operational distortion rate functions are given by (7.46).

We consider an arbitrary orthogonal transform matrix $\boldsymbol{A}_0$ and an arbitrary bit allocation given by the vector $\boldsymbol{b} = (R_0, \cdots R_{N-1})^T$ with $\sum_{i=0}^{N-1} R_i = R$. Starting with the given transform matrix $\boldsymbol{A}_0$ we apply an iterative algorithm that generates a sequence of orthonormal transform matrices $\{\boldsymbol{A}_k\}$. The corresponding autocorrelation matrices are given by $\boldsymbol{R}(\boldsymbol{A}_k) = \boldsymbol{A}_k \boldsymbol{R}_{SS} \boldsymbol{A}_k^T$ with $\boldsymbol{R}_{SS}$ denoting the autocorrelation matrix of the source signal. The transform coefficient variances $\sigma_i^2(\boldsymbol{A}_k)$ are the elements on the main diagonal of $\boldsymbol{R}(\boldsymbol{A}_k)$ and the distortion rate function for the transform coding system is given by

$$D(\boldsymbol{A}_k, R) = \sum_{i=0}^{N-1} \sigma_i^2(\boldsymbol{A}_k) \cdot g(R_i). \tag{7.47}$$

Each iteration $\boldsymbol{A}_k \mapsto \boldsymbol{A}_{k+1}$ shall consists of the following two steps:

(1) Consider the class of orthogonal reordering matrices $\{\boldsymbol{P}\}$, for which each row and column consists of a single one and $N-1$ zeros. The basis vectors given by the rows of $\boldsymbol{A}_k$ are reordered by a multiplication with the reordering matrix $\boldsymbol{P}_k$ that minimizes the distortion rate function $D(\boldsymbol{P}_k \boldsymbol{A}_k, R)$.

(2) Apply a Jacobi rotation[1] $\boldsymbol{A}_{k+1} = \boldsymbol{Q}_k(\boldsymbol{P}_k \boldsymbol{A}_k)$. The orthogonal matrix $\boldsymbol{Q}_k$ is determined in a way that the element r_{ij} on a secondary diagonal of $\boldsymbol{R}(\boldsymbol{P}_k \boldsymbol{A}_k)$ that has the largest absolute value becomes zero in $\boldsymbol{R}(\boldsymbol{A}_{k+1})$. $\boldsymbol{Q}_k$ is an elementary rotation matrix. It is an identity matrix where the main diagonal elements q_{ii} and q_{jj} are replaced by a value $\cos\varphi$ and the secondary diagonal elements q_{ij} and q_{ji} are replaced by the values $\sin\varphi$ and $-\sin\varphi$, respectively.

[1] The classical Jacobi algorithm [43, 18] for determining the eigendecomposition of real symmetric matrices consist of a sequence of Jacobi rotations.

It is obvious that the reordering step does not increase the distortion, i.e., $D(\boldsymbol{P}_k\boldsymbol{A}_k, R) \leq D(\boldsymbol{A}_k, R)$. Furthermore, for each pair of variances $\sigma_i^2(\boldsymbol{P}_k\boldsymbol{A}_k) \geq \sigma_j^2(\boldsymbol{P}_k\boldsymbol{A}_k)$, it implies $g(R_i) \leq g(R_j)$; otherwise, the distortion $D(\boldsymbol{P}_k\boldsymbol{A}_k, R)$ could be decreased by switching the i-th and j-th row of the matrix $\boldsymbol{P}_k\boldsymbol{A}_k$. A Jacobi rotation that zeros the element r_{ij} of the autocorrelation matrix $\boldsymbol{R}(\boldsymbol{P}_k\boldsymbol{A}_k)$ in $\boldsymbol{R}(\boldsymbol{A}_{k+1})$ does only change the variances for the i-th and j-th transform coefficient. If $\sigma_i^2(\boldsymbol{P}_k\boldsymbol{A}_k) \geq \sigma_j^2(\boldsymbol{P}_k\boldsymbol{A}_k)$, the variances are modified according to

$$\begin{aligned} \sigma_i^2(\boldsymbol{A}_{k+1}) &= \sigma_i^2(\boldsymbol{P}_k\boldsymbol{A}_k) + \delta(\boldsymbol{P}_k\boldsymbol{A}_k), && (7.48) \\ \sigma_j^2(\boldsymbol{A}_{k+1}) &= \sigma_j^2(\boldsymbol{P}_k\boldsymbol{A}_k) - \delta(\boldsymbol{P}_k\boldsymbol{A}_k), && (7.49) \end{aligned}$$

with

$$\delta(\boldsymbol{P}_k\boldsymbol{A}_k) = \frac{2r_{ij}^2}{(r_{ii} - r_{jj}) + \sqrt{(r_{ii} - r_{jj})^2 + 4r_{ij}^2}} \geq 0, \qquad (7.50)$$

and r_{ij} being the elements of the matrix $\boldsymbol{R}(\boldsymbol{P}_k\boldsymbol{A}_k)$. The overall distortion for the transform matrix $\boldsymbol{A}_{k+1}$ will never become smaller than the overall distortion for the transform matrix $\boldsymbol{A}_k$,

$$\begin{aligned} D(\boldsymbol{A}_{k+1}, R) &= \sum_{i=0}^{N-1} \sigma_i^2(\boldsymbol{A}_{k+1}) \cdot g(R_i) \\ &= D(\boldsymbol{P}_k\boldsymbol{A}_k, R) + \delta(\boldsymbol{P}_k\boldsymbol{A}_k) \cdot (g(R_i) - g(R_j)) \\ &\leq D(\boldsymbol{P}_k\boldsymbol{A}_k, R) \leq D(\boldsymbol{A}_k, R). && (7.51) \end{aligned}$$

The described algorithm represents the classical Jacobi algorithm [43, 18] with additional reordering steps. The reordering steps do not affect the basis vectors of the transform (rows of the matrices $\boldsymbol{A}_k$), but only their ordering. As the number of iteration steps approaches infinity, the transform matrix $\boldsymbol{A}_k$ approaches the transform matrix of a KLT and the autocorrelation matrix $\boldsymbol{R}(\boldsymbol{A}_k)$ approaches a diagonal matrix. Hence, for each possible bit allocation, there exists a KLT that gives an overall distortion that is smaller than or equal to the distortion for any other orthogonal transform. While the basis vectors of the transform are determined by the source signal, their ordering is determined by the relative ordering of the partial rates R_i inside the bit allocation vector $\boldsymbol{b}$ and the normalized operational distortion rate function $g(R_i)$.

We have shown that the KLT is the orthogonal transform that minimizes the distortion for a set of scalar quantizers that represent scaled versions of a given quantizer for unit variance. In particular, the KLT is the optimal transform for Gaussian sources if optimal scalar quantizers are used [19]. The KLT produces decorrelated transform coefficients. However, decorrelation does not necessarily imply independence. For non-Gaussian sources, other orthogonal transforms or nonorthogonal transforms can be superior with respect to the coding efficiency [20, 13].

Example for a Gauss-Markov Process. As an example, we consider the 3×3 KLT for a stationary Gauss-Markov process with zero mean, unit variance, and a correlation coefficient of $\rho = 0.9$. We assume a bit allocation vector $\boldsymbol{b} = [5, 3, 2]$ and consider entropy-constrained scalar quantizers. We further assume that the high-rate approximation of the operational distortion rate function $D_i(R_i) = \varepsilon^2\sigma_i^2 2^{-2R_i}$ with $\varepsilon^2 = \pi e/6$ is valid for the considered rates. The initial transform matrix $\boldsymbol{A}_0$ shall be the matrix of the DCT-II transform, which we will later introduce in sec. 7.5.3. The autocorrelation matrix $\boldsymbol{R}_{SS}$ and the initial transform matrix $\boldsymbol{A}_0$ are given by

$$\boldsymbol{R}_s = \begin{bmatrix} 1 & 0.9 & 0.81 \\ 0.9 & 1 & 0.9 \\ 0.81 & 0.9 & 1 \end{bmatrix}, \quad \boldsymbol{A}_0 = \begin{bmatrix} 0.5774 & 0.5774 & 0.5774 \\ 0.7071 & 0 & -0.7071 \\ 0.4082 & -0.8165 & 0.4082 \end{bmatrix}. \tag{7.52}$$

For the transform coefficients, we obtain the autocorrelation matrix

$$\boldsymbol{R}(\boldsymbol{A}_0) = \begin{bmatrix} 2.74 & 0 & -0.0424 \\ 0 & 0.19 & 0 \\ -0.0424 & 0 & 0.07 \end{bmatrix}. \tag{7.53}$$

The distortion $D(\boldsymbol{A}_0, R)$ for the initial transform is equal to 0.01426. We now investigate the effect of the first iteration of the algorithm described above. For the given relative ordering in the bit allocation vector $\boldsymbol{b}$, the optimal reordering matrix $\boldsymbol{P}_0$ is the identity matrix. The Jacobi rotation matrix $\boldsymbol{Q}_0$ and the resulting new transform matrix $\boldsymbol{A}_1$ are given by

$$\boldsymbol{Q}_0 = \begin{bmatrix} 0.9999 & 0 & -0.0159 \\ 0 & 1 & 0 \\ 0.0159 & 0 & 0.9999 \end{bmatrix}, \quad \boldsymbol{A}_1 = \begin{bmatrix} 0.5708 & 0.5902 & 0.5708 \\ 0.7071 & 0 & -0.7071 \\ 0.4174 & -0.8072 & 0.4174 \end{bmatrix}. \tag{7.54}$$

The parameter $\delta(\boldsymbol{P}_0\boldsymbol{A}_0)$ is equal to 0.000674. The distortion $D(\boldsymbol{A}_1, R)$ is equal to 0.01420. In comparison to the distortion for the initial transform matrix $\boldsymbol{A}_0$, it has been reduced by about 0.018 dB. The autocorrelation matrix $\boldsymbol{R}(\boldsymbol{A}_1)$ for the new transform coefficients is given by

$$\boldsymbol{R}(\boldsymbol{A}_1) = \begin{bmatrix} 2.7407 & 0 & 0 \\ 0 & 0.19 & 0 \\ 0 & 0 & 0.0693 \end{bmatrix}. \tag{7.55}$$

The autocorrelation matrix has already become a diagonal matrix after the first iteration. The transform given by $\boldsymbol{A}_1$ represents a KLT for the given source signal.

7.4.2 Asymptotic Operational Distortion Rate Function

In sec. 7.3.2, we considered the bit allocation for transform coding at high rates. An optimal bit allocation results in constant component distortions D_i, which are equal to the overall distortion D. By using the high rate approximation $D_i(R_i) = \varepsilon_i^2\ \sigma_i^2\ 2^{-2R_i}$ for the operational distortion rate function of the component quantizers, we derived the overall operational distortion rate function given in (7.36). For Gaussian sources and entropy-constrained scalar quantization, all parameters ε_i^2 are equal to $\varepsilon = \pi e/6$. And if we use a KLT of size N as transform matrix, the transform coefficient variances σ_i^2 are equal to the eigenvalues $\xi_i^{(N)}$ of the N-th order autocorrelation matrix $\boldsymbol{R}_N$ for the input process. Hence, for Gaussian sources and a transform coding system that consists of a KLT of size N and entropy-constrained scalar quantizers for the transform coefficients, the high rate approximation for the overall distortion rate function can be written as

$$D_N(R) = \frac{\pi\, e}{6} \left(\prod_{i=0}^{N-1} \xi_i^{(N)} \right)^{\frac{1}{N}} 2^{-2R}. \tag{7.56}$$

The larger we choose the transform size N of the KLT, the more the samples of the input source are decorrelated. For deriving a bound for the operational distortion rate function at high rates, we consider the limit for N approaching infinity. By applying Grenander and Szegö's theorem (4.76) for sequences of Toeplitz matrices, the limit of (7.56) for N approaching infinity can be reformulated using the power spectral

density $\Phi_{SS}(\omega)$ of the input source. For Gaussian sources, the asymptotic operational distortion rate function for high rates and large transform dimensions is given by

$$D_\infty(R) = \frac{\pi e}{6} \cdot 2^{\frac{1}{2\pi}\int_{-\pi}^{\pi} \log_2 \Phi_{SS}(\omega)\, \mathrm{d}\omega} \cdot 2^{-2R}. \tag{7.57}$$

A comparison with the Shannon lower bound (4.77) for zero-mean Gaussian sources shows that the asymptotic operational distortion rate function lies 1.53 dB or 0.25 bit per sample above this fundamental bound. The difference is equal to the space-filling advantage of high-dimensional vector quantization. For zero-mean Gaussian sources and high rates, the memory and shape advantage of vector quantization can be completely exploited using a high-dimensional transform coding.

By using the relationship $\sigma_S^2 = \frac{1}{2\pi}\int_{-\pi}^{\pi} \Phi_{SS}(\omega)\mathrm{d}\omega$ for the variance of the input source, the asymptotic transform coding gain for zero-mean Gaussian sources can be expressed as the ratio of the arithmetic and geometric means of the power spectral density,

$$G_T^\infty = \frac{\varepsilon^2 \sigma_S^2 2^{-2R}}{D_\infty(R)} = \frac{\frac{1}{2\pi}\int_{-\pi}^{\pi} \Phi_{SS}(\omega) d\omega}{2^{\frac{1}{2\pi}\int_{-\pi}^{\pi} \log_2 \Phi_{SS}(\omega) d\omega}} \tag{7.58}$$

The asymptotic transform coding gain at high rates is identical to the approximation for the DPCM coding gain at high rates (6.75).

Zero-Mean Gauss-Markov Sources. We now consider the special case of zero-mean Gauss-Markov sources. The product of the eigenvalues $\xi_i^{(N)}$ of a matrix $\boldsymbol{R}_N$ is always equal to the determinant $|\boldsymbol{R}_N|$ of the matrix. And for zero-mean sources, the N-th order autocorrelation matrix $\boldsymbol{R}_N$ is equal to the N-th order autocovariance matrix $\boldsymbol{C}_N$. Hence, we can replace the product of the eigenvalues in (7.56) with the determinant $|\boldsymbol{C}_N|$ of the N-th order autocovariance matrix. Furthermore, for Gauss-Markov sources, the determinant of the N-th order autocovariance matrix can be expressed according to (2.50). Using these relationships, the operational distortion rate function for zero-mean Gauss-Markov sources and a transform coding system with an N-dimensional KLT and entropy-constrained component quantizers is given by

$$D_N(R) = \frac{\pi\, e}{6}\, \sigma_S^2\, (1-\rho^2)^{\frac{N-1}{N}}\, 2^{-2R}, \tag{7.59}$$

where σ_S^2 and ρ denote the variance and the correlation coefficient of the input source. For the corresponding transform gain, we obtain

$$G_T^N = (1-\rho^2)^{\frac{1-N}{N}}. \tag{7.60}$$

The asymptotic operational distortion rate function and the asymptotic transform gain for high rates and N approaching infinity are given by

$$D_\infty(R) = \frac{\pi\, e}{6}\, \sigma_S^2\, (1-\rho^2)\, 2^{-2R}, \qquad G_T^\infty = \frac{1}{(1-\rho^2)}. \tag{7.61}$$

7.4.3 Performance for Gauss-Markov Sources

For demonstrating the effectiveness of transform coding for correlated input sources, we used a Gauss-Markov source with zero mean, unit variance, and a correlation coefficient of $\rho = 0.9$ and compared the rate distortion efficiency of transform coding with KLT's of different sizes N and entropy-constrained scalar quantization (ECSQ) with the fundamental rate distortion bound and the rate distortion efficiency for ECSQ of the input samples. The experimentally obtained data and

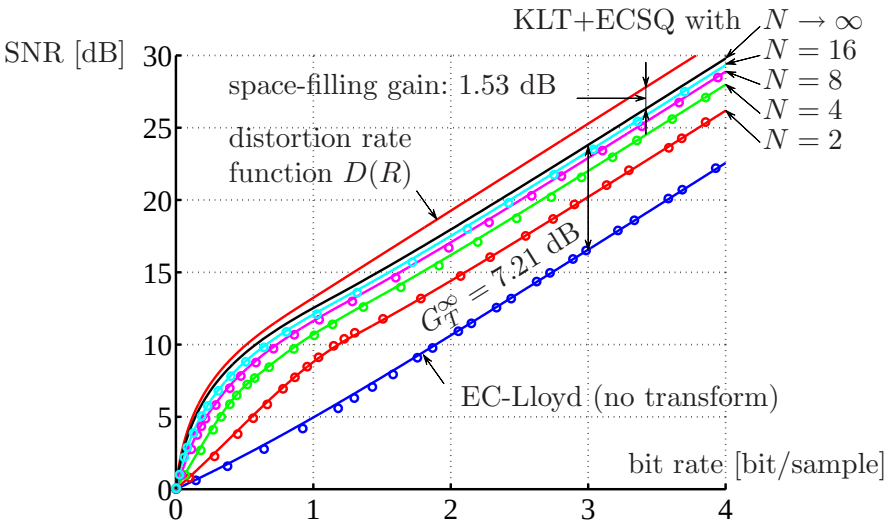


Fig. 7.6 Transform coding of a Gauss-Markov source with zero mean, unit variance, and a correlation coefficient of $\rho = 0.9$. The diagram compares the efficiency of direct ECSQ and transform coding with ECSQ to the distortion rate function $D(R)$. The circles represent experimental data while the solid lines represent calculated curves. The rate is measured as the average of the entropies for the outputs of the component quantizers.

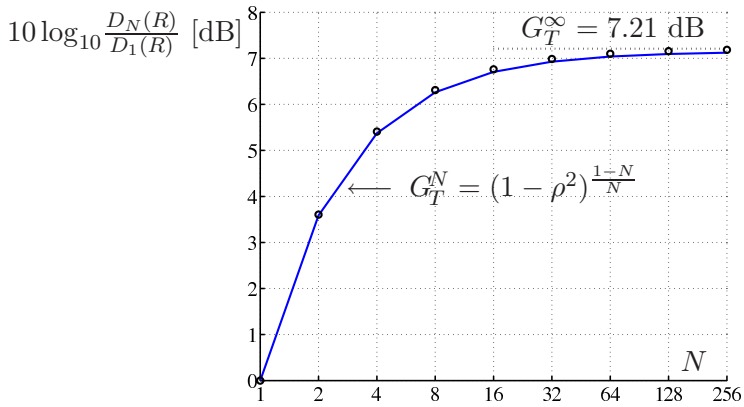


Fig. 7.7 Transform gain as function of the transform size N for a zero-mean Gauss-Markov source with a correlation factor of $\rho = 0.9$.

the calculated distortion rate curves are shown in Fig. 7.6. The rate was determined as average of the entropies of the quantizer outputs. It can be seen that transform coding significantly increases the coding efficiency relative to direct ECSQ. An interesting fact is that for transform sizes larger than $N = 4$ the distance to the fundamental rate distortion bound at low rates is less than at high rates. A larger transform size N generally yields a higher coding efficiency. However, the asymptotic bound (7.61) is already nearly achieved for a moderate transform size of $N = 16$ samples. A further increase of the transform size N would only slightly improve the coding efficiency for the example source. This is further illustrated in Fig. 7.7, which shows the transform coding gain as function of the transform size N.

7.5 Signal-Independent Unitary Transforms

Although the KLT has several desirable properties, it is not used in practically video coding applications. One of the reasons is that there are no fast algorithms for calculating the transform coefficients for a general KLT. Furthermore, since the KLT is signal-dependent, a single transform matrix is not suitable for all video sequences, and adaptive schemes are only implementable at an additional computational complexity. In the following, we consider signal-independent transforms. The transform that is used in all practically used video coding

schemes is the discrete cosine transform (DCT), which will be discussed in sec. 7.5.3. In addition, we will briefly review the Walsh-Hadamard transform and, for motivating the DCT, the discrete Fourier transform.

7.5.1 The Walsh-Hadamard Transform (WHT)

The Walsh-Hadamard transform is a very simple orthogonal transform that can be implemented using only additions and a final scaling. For transform sizes N that represent positive integer power of 2, the transform matrix $\boldsymbol{A}_N$ is recursively defined by

$$\boldsymbol{A}_N = \frac{1}{\sqrt{2}} \begin{bmatrix} \boldsymbol{A}_{N/2} & \boldsymbol{A}_{N/2} \\ \boldsymbol{A}_{N/2} & -\boldsymbol{A}_{N/2} \end{bmatrix} \qquad \text{with} \qquad \boldsymbol{A}_1 = [1]. \tag{7.62}$$

When ignoring the constant normalization factor, the Hadamard transform matrices only consist of entries equal to 1 and -1 and, hence, the transform coefficients can be calculated very efficiently. However, due to its piecewise-constant basis vectors, the Hadamard transform produces subjectively disturbing artifacts if it is combined with strong quantization of the transform coefficients. In video coding, the Hadamard transform is only used for some special purposes. An example is the second-level transform for chroma coefficients in H.264/AVC [36]

7.5.2 The Discrete Fourier Transform (DFT)

One of the most important transforms in communications engineering and signal processing is the Fourier transform. For discrete-time signals of a finite length N, the discrete Fourier transform (DFT) is given by

$$u[k] = \frac{1}{\sqrt{N}} \sum_{n=0}^{N-1} s[n]\, e^{-j\frac{2\pi kn}{N}}, \tag{7.63}$$

where $s[n]$, with $0 \leq n < N$, and $u[k]$, with $0 \leq k < N$, represent the components of the signal vector $\boldsymbol{s}$ and the vector of transform coefficients $\boldsymbol{u}$, respectively, and j is the imaginary unit. The inverse DFT is given by

$$s[n] = \frac{1}{\sqrt{N}} \sum_{k=0}^{N-1} u[k]\, e^{j\frac{2\pi kn}{N}}. \tag{7.64}$$

For computing both the forward and inverse transform fast algorithms (FFT) exist, which use sparse matrix factorization. The DFT generally produces complex transform coefficients. However, for real input signals, the DFT obeys the symmetry $u[k] = u^*[N-k]$, where the asterisk denotes complex conjugation. Hence, an input signal of N real samples is always completely specified by N real coefficients.

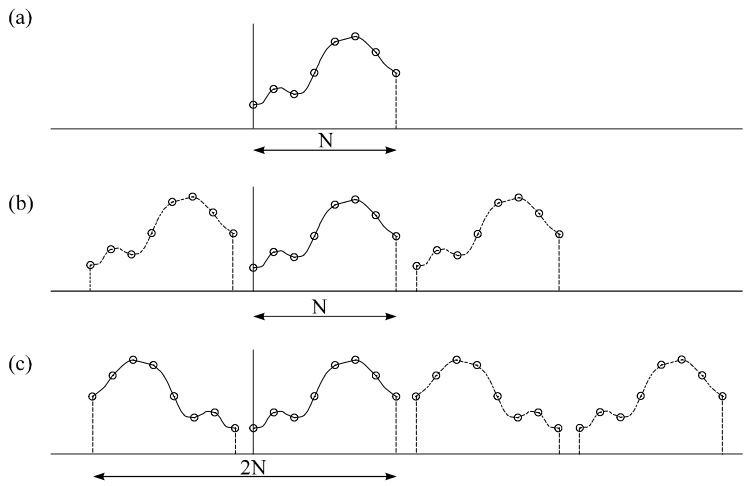


Fig. 7.8 Periodic signal extensions for the DFT and the DCT: (a) input signal; (b) signal replica for the DFT; (c) signal replica for the DCT-II.

The discrete Fourier transform is rarely used in compression systems. One reason is its complex nature. Another reason is the fact that the DFT implies a periodic signal extension. The basis functions of the DFT are complex exponentials, which are periodic functions. For each basis function, a particular integer multiple of the period is equal to the length of the input signal. Hence, the signal that is actually represented by the DFT coefficients is a periodically extended version of the finite-length input signal, as illustrated in Fig. 7.8. Any discontinuity between the left and right signal boundary reduces the rate of convergence of the Fourier series, i.e., more basis functions are needed to represent the input signal with a given accuracy. In combination with strong quantization this leads also to significant high-frequent artifacts in the reconstruction signal.

7.5.3 The Discrete Cosine Transform (DCT)

The magnitudes of the high-frequency DFT coefficients can be reduced by symmetrically extending the finite-length input signal at its boundaries and applying a DFT of approximately double size. If the extended signal is mirror symmetric around the origin, the imaginary sine terms get eliminated and only real cosine terms remain. Such a transform is denoted as discrete cosine transform (DCT). There are several DCT's, which differ in the introduced signal symmetry. The most commonly used form is the DCT-II, which can be derived by introducing mirror symmetry with sample repetition at both boundaries as illustrated in Fig. 7.8(c). For obtaining mirror symmetry around the origin, the signal has to be shifted by half a sample. The signal s' of $2N$ samples that is actually transformed using the DFT is given by

$$s'[n] = \begin{cases} s[n-1/2] & : \quad 0 \leq n < N \\ s[2N-n-3/2] & : \quad N \leq n < 2N \end{cases} . \tag{7.65}$$

For the transform coefficients $u'[k]$, we obtain

$$\begin{aligned} u'[k] &= \frac{1}{\sqrt{2N}} \sum_{i=0}^{2N-1} s'[i] e^{-j\frac{2\pi kn}{2N}} \\ &= \frac{1}{\sqrt{2N}} \sum_{n=0}^{N-1} s[n-1/2] \left(e^{-j\frac{\pi}{N}kn} + e^{-j\frac{\pi}{N}k(2N-n-1)} \right) \\ &= \frac{1}{\sqrt{2N}} \sum_{n=0}^{N-1} s[n] \left(e^{-j\frac{\pi}{N}k\left(n+\frac{1}{2}\right)} + e^{j\frac{\pi}{N}k\left(n+\frac{1}{2}\right)} \right) \\ &= \sqrt{\frac{2}{N}} \sum_{n=0}^{N-1} s[n] \cos\left(\frac{\pi}{N} k \left(n + \frac{1}{2} \right) \right) . \end{aligned} \tag{7.66}$$

In order to get an orthogonal transform, the DC coefficient $u'[0]$ has to be divided by $\sqrt{2}$. The forward transform of the DCT-II is given by

$$u[k] = \sum_{n=0}^{N-1} s[n]\, \alpha_k \cos\left(\frac{\pi}{N} k \left(n + \frac{1}{2} \right) \right) , \tag{7.67}$$

with

$$\alpha_n = \sqrt{\frac{1}{N}} \cdot \begin{cases} 1 & : \quad n = 0 \\ \sqrt{2} & : \quad n > 0 \end{cases} . \tag{7.68}$$

The inverse transform is given by

$$s[n] = \sum_{k=0}^{N-1} u[k] \cdot \alpha_k \cdot \cos\left(\frac{\pi}{N} k \left(n + \frac{1}{2}\right)\right). \tag{7.69}$$

The DCT-II is the most commonly used transform in image and video coding application. It is included in the following coding standards: JPEG [38], H.261 [35], H.262/MPEG-2 [39], H.263 [36], and MPEG-4 [31]. Although, the most recent video coding standard H.264/AVC [36] does not include a DCT as discussed above, it includes an integer approximation of the DCT that has similar properties, but can be implemented more efficiently and does not cause an accumulation of rounding errors inside the motion-compensation loop. The justification for the wide usage of the DCT includes the following points:

- The DCT does not depend on the input signal.
- There are fast algorithms for computing the forward and inverse transform.
- The DCT can be extended to two (or more) dimensions in a separable way.
- The DCT is a good approximation of the KLT for highly correlated Gauss-Markov sources (see below).

Comparison of DCT and KLT. In contrast to the KLT, the basis vectors of the DCT are independent of the input source and there exist fast algorithms for computing the forward and inverse transforms. For zero-mean Gauss-Markov sources with large correlation coefficients ρ, the DCT-II basis vectors are a good approximation of the eigenvectors of the autocorrelation matrix $\boldsymbol{R}_{SS}$. If we neglect possible multiplications with -1, the basis vectors of the KLT for zero-mean Gauss-Markov sources approach the DCT-II basis vectors as the correlation coefficient ρ approaches one [2]. This is illustrated in Fig. 7.9. On the left side of this figure, the basis vectors of a KLT for zero-mean Gauss-Markov sources with a correlation coefficient of $\rho = 0.9$ are compared with the basis vectors of the DCT-II. On the right side of Fig. 7.9, the mean square difference $\delta(\rho)$ between the transform matrix of the

DCT-II $\boldsymbol{A}_{\mathrm{DCT}}$ and the KLT transform matrix $\boldsymbol{A}_{\mathrm{KLT}}$ is shown as function of the correlation coefficient ρ. For this experiment, we used the KLT transform matrices $\boldsymbol{A}_{\mathrm{KLT}}$ for which the basis vectors (rows) are ordered in decreasing order of the associated eigenvalues and all entries in the first column are non-negative.

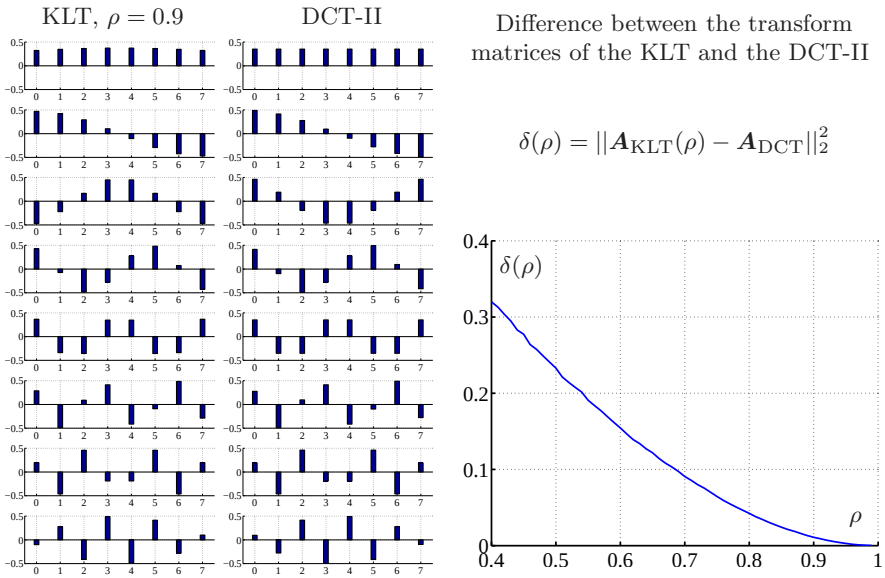


Fig. 7.9 Comparison of the basis vectors of the DCT-II and the KLT for zero-mean Gauss-Markov sources for a transform size $N = 8$: (left) basis vectors of the DCT-II and a KLT for $\rho = 0.9$; (right) mean square difference between the DCT-II and the KLT transform matrix as function of the correlation coefficient ρ.

7.6 Transform Coding Example

As a simple transform coding example, we consider the Hadamard transform of the size $N = 2$ for a zero-mean Gauss-Markov process with a variance σ_S^2 and a correlation coefficient ρ. The input vectors $\boldsymbol{s}$ and the orthogonal analysis transform matrix $\boldsymbol{A}$ are given by

$$\boldsymbol{s} = \begin{bmatrix} s_0 \\ s_1 \end{bmatrix} \quad \text{and} \quad \boldsymbol{A} = \frac{1}{\sqrt{2}} \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix}. \tag{7.70}$$

The analysis transform

$$\boldsymbol{u} = \begin{bmatrix} u_0 \\ u_1 \end{bmatrix} = \boldsymbol{A}\boldsymbol{s} = \frac{1}{\sqrt{2}} \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix} \begin{bmatrix} s_0 \\ s_1 \end{bmatrix} \tag{7.71}$$

yields the transform coefficients

$$u_0 = \frac{1}{\sqrt{2}}\,(s_0 + s_1), \qquad u_0 = \frac{1}{\sqrt{2}}\,(s_0 - s_1). \tag{7.72}$$

For the Hadamard transform, the synthesis transform matrix $\boldsymbol{B}$ is equal to the analysis transform matrix, $\boldsymbol{B} = \boldsymbol{A}^T = \boldsymbol{A}$.

The transform coefficient variances are given by

$$\begin{aligned} \sigma_0^2 &= E\{U_0^2\} = E\left\{\frac{1}{2}(S_0 + S_1)^2\right\} \\ &= \frac{1}{2}\left(E\{S_0^2\} + E\{S_1^2\} + 2\,E\{S_0 S_1\}\right) \\ &= \frac{1}{2}\left(\sigma_S^2 + \sigma_S^2 + 2\,\sigma_S^2\,\rho\right) = \sigma_S^2(1+\rho), \qquad (7.73) \\ \sigma_{u_1}^2 &= E\{U_1^2\} = \sigma_S^2(1-\rho), \qquad (7.74) \end{aligned}$$

where S_i and U_i denotes the random variables for the signal components and transform coefficients, respectively. The cross-correlation of the transform coefficients is

$$\begin{aligned} E\{U_0 U_1\} &= \frac{1}{2}\,E\{(S_0 + S_1)(S_0 - S_1)\} \\ &= \frac{1}{2}\,E\{(S_0^2 - S_1^2)\} = \frac{1}{2}(\sigma_S^2 - \sigma_S^2) = 0. \qquad (7.75) \end{aligned}$$

The Hadamard transform of size $N = 2$ generates independent transform coefficients for zero-mean Gauss-Markov sources. Hence, it is a KLT for all correlation coefficients ρ. It is also the DCT-II for $N = 2$.

In the following, we consider entropy-constrained scalar quantization of the transform coefficients at high rates. The high-rate approximation of the operational distortion rate function for entropy-constrained scalar quantization of Gaussian sources is given by $D_i(R_i) = \varepsilon^2 \sigma_i^2 2^{-2R_i}$ with $\varepsilon^2 = \pi e/6$. The optimal bit allocation rule

for high rates (cp. sec. 7.3.2) yields the component rates

$$R_0 = R + \frac{1}{4}\log_2\left(\frac{1+\rho}{1-\rho}\right), \tag{7.76}$$

$$R_1 = R - \frac{1}{4}\log_2\left(\frac{1+\rho}{1-\rho}\right), \tag{7.77}$$

where R denotes the overall rate. If $\rho > 0$, the rate R_0 for the DC coefficient u_0 is always $\frac{1}{2}\log_2\left(\frac{1+\rho}{1-\rho}\right)$ bits larger than the rate R_1 for the AC coefficient u_1. The high-rate operational distortion rate function for the considered transform coder is given by

$$D(R) = \varepsilon^2\sigma_S^2\sqrt{1-\rho^2}\cdot 2^{-2R}. \tag{7.78}$$

A comparison with the Shannon Lower bound (4.80) shows that, for high rates, the loss against the fundamental rate distortion bound is

$$\frac{D(R)}{D_L(R)} = \frac{\pi\, e}{6\sqrt{1-\rho^2}}. \tag{7.79}$$

For zero-mean Gauss-Markov sources with $\rho = 0.9$ and high rates, the transform coding gain is about 3.61 dB, while the loss against the Shannon lower bound is about 5.14 dB. The transform coding gain can be increased by applying larger decorrelating transforms.

7.7 Summary of Transform Coding

In this chapter, we discussed transform coding with orthogonal block transforms. An orthogonal block transform of size N specifies a rotation or reflection of the coordinate system in the N-dimensional signal space. We showed that a transform coding system with an orthogonal block transform and scalar quantization of the transform coefficients represents a vector quantizer for which the quantization cells are hyperrectangles in the N-dimensional signal space. In contrast to scalar quantization in the original domain, the grid of quantization cells is not aligned with the coordinate axes of the original space. A decorrelation transform rotates the coordinate system toward the primary axes of the N-dimensional joint pdf, which has the effect that, for correlated sources, scalar quantization in the transform domain becomes more effective than in the original signal space.

The optimal distribution of the overall bit rate among the transform coefficients was discussed in some detail with the emphasis on Gaussian sources and high rates. In general, an optimal bit allocation is obtained if all component quantizers are operated at the same slope of their operational distortion rate functions. For high rates, this is equivalent to a bit allocation that yields equal component distortions. For stationary sources with memory the effect of the unitary transform is a nonuniform assignment of variances to the transform coefficients. This nonuniform distribution is the reason for the transform gain in case of optimal bit allocation.

The KLT was introduced as the transform that generates decorrelated transform coefficients. We have shown that the KLT is the optimal transform for Gaussian sources if we use the same type of optimal quantizers, with appropriately scaled reconstruction levels and decision thresholds, for all transform coefficients. For the example of Gaussian sources, we also derived the asymptotic operational distortion rate function for large transform sizes and high rates. It has been shown that, for zero-mean Gaussian sources and entropy-constrained scalar quantization, the distance of the asymptotic operational distortion rate function to the fundamental rate distortion bounds is basically reduced to the space-filling advantage of vector quantization.

In practical video coding systems, KLT's are not used, since they are signal-dependent and cannot be implemented using fast algorithms. The most widely used transform is the DCT-II, which can be derived from the discrete Fourier transform (DFT) by introducing mirror symmetry with sample repetition at the signal boundaries and applying a DFT of double size. Due to the mirror symmetry, the DCT significantly reduces the blocking artifacts compared to the DFT. For zero-mean Gauss-Markov sources, the basis vectors of the KLT approach the basis vectors of the DCT-II as the correlation coefficient approaches one.

For highly-correlated sources, a transform coding system with a DCT-II and entropy-constrained scalar quantization of the transform coefficients is highly efficient in terms of both rate distortion performance and computational complexity.

8

Summary

The problem of communication may be posed as conveying source data with the highest fidelity possible without exceeding an available bit rate, or it may be posed as conveying the source data using the lowest bit rate possible while maintaining a specified reproduction fidelity. In either case, a fundamental trade-off is made between bit rate and signal fidelity. Source coding as described in this text provides the means to effectively control this trade-off.

Two types of source coding techniques are typically named: lossless and lossy coding. The goal of lossless coding is to reduce the average bit rate while incurring no loss in fidelity. Lossless coding can provide a reduction in bit rate compared to the original data, when the original signal contains dependencies or statistical properties that can be exploited for data compaction. The lower bound for the achievable bit rate of a lossless code is the discrete entropy rate of the source. Techniques that attempt to approach the entropy limit are called entropy coding algorithms. The presented entropy coding algorithms include Huffman codes, arithmetic codes, and the novel PIPE codes. Their application to discrete sources with and without consideration of statistical dependencies inside a source is described.

The main goal of lossy coding is to achieve lower bit rates than with lossless coding techniques while accepting some loss in signal fidelity. Lossy coding is the primary coding type for the compression of speech, audio, picture, and video signals, where an exact reconstruction of the source data is often not required. The fundamental limit for lossy coding algorithms is given by the rate distortion function, which specifies the minimum bit rate that is required for representing a source without exceeding a given distortion. The rate distortion function is derived as a mathematical function of the input source, without making any assumptions about the coding technique.

The process of incurring a reduction of signal fidelity is called quantization. Quantizers allow to effectively trade-off bit rate and signal fidelity and are at the core of every lossy source coding system. They can be classified into scalar and vector quantizers. For data containing none or little statistical dependencies, the combination of scalar quantization and scalar entropy coding is capable of providing a high coding efficiency at a low complexity level.

When the input data contain relevant statistical dependencies, these can be exploited via various techniques that are applied prior to or after scalar quantization. Prior to scalar quantization and scalar entropy coding, the statistical dependencies contained in the signal can be exploited through prediction or transforms. Since the scalar quantizer performance only depends on the marginal probability distribution of the input samples, both techniques, prediction and transforms, modify the marginal probability distribution of the samples to be quantized, in comparison to the marginal probability distribution of the input samples, via applying signal processing to two or more samples.

After scalar quantization, the applied entropy coding method could also exploit the statistical dependencies between the quantized samples. When the high rate assumptions are valid, it has been shown that this approach achieves a similar level of efficiency as techniques applied prior to scalar quantization. Such advanced entropy coding techniques are, however, associated with a significant complexity and, from practical experience, they appear to be inferior in particular at low bit rates.

The alternative to scalar quantization is vector quantization. Vector quantization allows the exploitation of statistical dependencies within

the data without the application of any signal processing algorithms in advance of the quantization process. Moreover, vector quantization offers a benefit that is unique to this techniques as it is a property of the quantization in high-dimensional spaces: the space filling advantage. The space filling advantage is caused by the fact that a partitioning of high-dimensional spaces into hyperrectangles, as achieved by scalar quantization, does not represent the densest packing. However, this gain can be only achieved by significantly increasing the complexity in relation to scalar quantization. In practical coding systems, the space filling advantage is usually ignored. Vector quantization is typically only used with certain structural constraints, which significantly reduce the associated complexity.

The present text describes the subject of source coding for 1-d discrete-time signals. For the quantitative analysis of the efficiency of the presented coding techniques, the source signals are considered as realizations of simple stationary random processes. The second part of the monograph discusses the subject of video coding. There are several important differences between source coding of 1-d stationary model sources and the compression of natural camera-view video signals. The first and most obvious difference is that we move from 1-d to 2-d signals in case of picture coding and to 3-d signals in case of video coding. Hence, the 1-d concepts need to be extended accordingly. Another important difference is that the statistical properties of natural camera-view video signals are nonstationary and, at least to a significant extend, unknown in advance. For an efficient coding of video signals, the source coding algorithms need to be adapted to the local statistics of the video signal as we will discuss in the second part of this monograph.

Acknowledgements

This text is based on a lecture that was held by one of us (T.W.) at the Berlin Institute of Technology during 2008-2010. The original lecture slides were inspired by lectures of Bernd Girod, Thomas Sikora, and Peter Noll as well as tutorial slides of Robert M. Gray. These individuals are greatly acknowledged for the generous sharing of their course material.

In the preparation of the lecture, Haricharan Lakshman was of exceptional help and his contributions are hereby acknowledged. We also want to thank Detlev Marpe, Gary J. Sullivan, and Martin Winken for the many helpful discussions on various subjects covered in the text that led to substantial improvements.

The impulse toward actually turning the lecture slides into the present monograph was given by Robert M. Gray, Editor-in-Chief of Now Publisher's *Foundations and Trends in Signal Processing*, through his invitation to write this text. During the lengthy process of writing, his and the anonymous reviewers' numerous valuable and detailed comments and suggestions greatly improved the final result.

The authors would also like to thank their families and friends for their patience and encouragement to write this monograph.

References

[1] N. M. Abramson. *Information Theory and Coding*. McGraw-Hill, New York, NY, USA, 1963.

[2] N. Ahmed, T. Natarajan, and K. R. Rao. Discrete Cosine Transform. *IEEE Transactions on Computers*, 23(1):90–93, 1974.

[3] S. Arimoto. An Algorithm for Calculating the Capacity of an Arbitrary Discrete Memoryless Channel. *IEEE Transactions on Information Theory*, 18:14–20, Jan. 1972.

[4] T. Berger. *Rate Distortion Theory*. Prentice-Hall, Englewood Cliffs, NJ, USA, 1971.

[5] J. Binia, M. Zakai, and J. Ziv. On the ϵ-Entropy and the Rate-Distortion Function of Certain Non-Gaussian Processes. *IEEE Transactions on Information Theory*, 20:514–524, July 1974.

[6] R. E. Blahut. Computation of Channel Capacity and Rate-Distortion Functions. *IEEE Transactions on Information Theory*, 18:460–473, Apr. 1972.

[7] M. Burrows and D. Wheeler. A Block-Sorting Lossless Data Compression Algorithm. Research Report 124, Digital Equipment Corporation, Palo Alto, CA, USA, May 1994.

[8] P.-C. Chang and R. M. Gray. Gradient Algorithms for Designing Predictive Vector Quantizers. *IEEE Transactions on Acoustics, Speech and Signal Processing*, 34(4):679–690, Aug. 1986.

[9] P. A. Chou, T. Lookabaugh, and R. M. Gray. Entropy-Constrained Vector Quantization. *IEEE Transactions on Acoustics, Speech and Signal Processing*, 37(1):31–42, Jan. 1989.

[10] R. J. Clarke. *Transform Coding of Images*. Academic Press, Orlando, Fla., 1985.

[11] T. M. Cover and J. A. Thomas. *Elements of Information Theory*. John Wiley and Sons, Hoboken, NJ, USA, 2006. 2nd edition.

[12] R. D. Dony and S. Haykin. Optimally Adaptive Transform Coding. *IEEE Transactions on Image Processing*, 4(10):1358–1370, Oct. 1995.

[13] M. Effros, H. Feng, and K. Zeger. Suboptimality of the Karhunen-Loève Transform for Transform Coding. *IEEE Transactions on Information Theory*, 50(8):1605–1619, Aug. 2004.

[14] R. G. Gallager. *Information Theory and Reliable Communication*. John Wiley & Sons, New York, USA, 1968.

[15] R. G. Gallager. Variations on a Theme by Huffman. *IEEE Transactions on Information Theory*, 24(6):668–674, Nov. 1978.

[16] A. Gersho and R. M. Gray. *Vector Quantization and Signal Compression*. Kluwer Academic Publishers, Boston, Dordrecht, London, 1992.

[17] H. Gish and J. N. Pierce. Asymptotically Efficient Quantizing. *IEEE Transactions on Information Theory*, 14:676–683, Sept. 1968.

[18] G. H. Golub and H. A. van der Vorst. Eigenvalue Computation in the 20th Century. *Journal of Computational and Applied Mathematics*, 123:35–65, 2000.

[19] V. K. Goyal. High-Rate Transform Coding: How High is High, and Does it Matter? In *Proceedings of the IEEE International Symposium on Information Theory*, Sorento, Italy, June 2000.

[20] V. K. Goyal. Theoretical Foundations of Transform Coding. *IEEE Signal Processing Magazine*, 18(5):9–21, Sept. 2001.

[21] V. K. Goyal, J. Zhuang, and M. Vetterli. Transform Coding with Backward Adaptive Updates. *IEEE Transactions on Information Theory*, 46(4):1623–1633, July 2000.

[22] R. M. Gray. *Source Coding Theory*. Kluwer Academic Publishers, Norwell, MA, USA, 1990.

[23] R. M. Gray. Toeplitz and Circulant Matrices: A Review. *Foundations and Trends in Communication and Information Theory*, 2(3):155–329, 2005.

[24] R. M. Gray. *Linear Predictive Coding and the Internet Protocol*. Now Publishers Inc, Boston-Delft, 2010.

[25] R. M. Gray and L. D. Davisson. *Random Processes: A Mathematical Approach for Engineers*. Prentice Hall, Englewood Cliffs, NJ, USA, 1985.

[26] R. M. Gray and L. D. Davisson. *An Introduction to Statistical Signal Processing*. Cambridge University Press, 2004.

[27] R. M. Gray and A. H. Gray. Asymptotically Optimal Quantizers. *IEEE Transactions on Information Theory*, 23:143–144, Jan. 1977.

[28] R. M. Gray and D. L. Neuhoff. Quantization. *IEEE Transactions on Information Theory*, 44(6):2325–2383, Oct. 1998.

[29] U. Grenander and G. Szegö. *Toeplitz Forms and Their Applications*. University of California Press, Berkeley and Los Angeles, USA, 1958.

[30] D. A. Huffman. A Method for the Construction of Minimum Redundancy Codes. *Proc. IRE*, pages 1098–1101, Sept. 1952.

[31] ISO/IEC. Coding of Audio-Visual Objects — Part 2: Visual. ISO/IEC 14496-2, Apr. 1999.

[32] ITU-T. Narrow-Band Visual Telephone Systems and Terminal Equipment. ITU-T Rec. H.320, 1999.
[33] ITU-T. Packet-Based Multimedia Communications Systems. ITU-T Rec. H.323, 1998.
[34] ITU-T. Terminal for Low Bit Rate Multimedia Communication. ITU-T Rec. H.324, 1996.
[35] ITU-T. Video Codec for Audiovisual Services at $p \times 64$ kbit/s. ITU-T Rec. H.261, Mar. 1993.
[36] ITU-T and ISO/IEC. Advanced Video Coding for Generic Audiovisual Services. ITU-T Rec. H.264 and ISO/IEC 14496-10 (MPEG-4 AVC), Mar. 2010.
[37] ITU-T and ISO/IEC. Generic Coding of Moving Pictures and Asociated Audio Information — Part 1: Systems. ITU-T Rec. H.222.0 and ISO/IEC 13818-1 (MPEG-2 Systems), Nov. 1994.
[38] ITU-T and ISO/IEC. Digital Compression and Coding of Continuous-Tone Still Images. ITU-T Rec. T.81 and ISO/IEC 10918-1 (JPEG), Sept. 1992.
[39] ITU-T and ISO/IEC. Generic Coding of Moving Pictures and Associated Audio Information — Part 2: Video. ITU-T Rec. H.262 and ISO/IEC 13818-2, Nov. 1994.
[40] ITU-T and ISO/IEC. Lossless and Near-Lossless Compression of Continuous-Tone Still Images. ITU-T Rec. T.87 and ISO/IEC 14495-1 (JPEG-LS), June 1998.
[41] ITU-T and ISO/IEC. JPEG 2000 Image Coding System – Core Coding System. ITU-T Rec. T.800 and ISO/IEC 15444-1 (JPEG 2000), 2002.
[42] ITU-T and ISO/IEC. JPEG XR Image Coding System – Image Coding Specification. ITU-T Rec. T.832 and ISO/IEC 29199-2 (JPEG XR), 2009.
[43] C. G. J. Jacobi. Über ein leichtes Verfahren, die in der Theorie der Säcularströmungen vorkommenden Gleichungen numerisch aufzulösen. *Journal für reine und angewandte Mathematik*, 30:51–94, 1846.
[44] N. S. Jayant and P. Noll. *Digital Coding of Waveforms*. Prentice-Hall, Englewood Cliffs, NJ, USA, 1994.
[45] A. N. Kolmogorov. *Grundbegriffe der Wahrscheinlichkeitsrechnung*. Springer, Berlin, Germany, 1933. An English translation by N. Morrison appeared under the title Foundations of the Theory of Probability (Chelsea, New York) in 1950, with a second edition in 1956.
[46] Y. Linde, A. Buzo, and R. M. Gray. An Algorithm for Vector Quantizer Design. *IEEE Transactions on Communications*, 28(1):84–95, Jan. 1980.
[47] T. Linder and R. Zamir. On the Asymptotic Tightness of the Shannon Lower Bound. *IEEE Transactions on Information Theory*, 40(6):2026–2031, Nov. 1994.
[48] Y. N. Linkov. Evaluation of Epsilon Entropy of Random Variables for Small Esilon. *Problems of Information Transmission*, 1:12–18, 1965.
[49] S. P. Lloyd. Least Squares Quantization in PCM. *IEEE Transactions on Information Theory*, 28:127–135, Mar. 1982. Unpublished Bell Laboratories Technical Note, 1957.

[50] T. D. Lookabaugh and R. M. Gray. High-Resolution Quantization Theory and the Vector Quantizer Advantage. *IEEE Transactions on Information Theory*, 35(5):1020–1033, Sept. 1989.

[51] J. Makhoul. Linear Prediction: A Tutorial Review. *Proceedings of the IEEE*, 63(4):561–580, Apr. 1975.

[52] J. Makhoul, S. Roucos, and H. Gish. Vector Quantization in Speech Coding. *Proceedings of the IEEE*, 73(11):1551–1587, Nov. 1985.

[53] H. S. Malvar. *Signal Processing with Lapped Transforms*. Artech House, Norwood, MA, USA, 1992.

[54] D. Marpe, H. Schwarz, and T. Wiegand. Probability Interval Partitioning Entropy Codes. Submitted to IEEE Transactions on Information Theory, 2010. Available at http://iphome.hhi.de/marpe/download/pipe-subm-ieee10.pdf.

[55] D. Marpe, H. Schwarz, and T. Wiegand. Context-Adaptive Binary Arithmetic Coding for H.264/AVC. *IEEE Transactions on Circuits and Systems for Video Technology*, 13(7):620–636, July 2003.

[56] J. Max. Quantizing for Minimum Distortion. *IRE Transactions on Information Theory*, 6(1):7–12, Mar. 1960.

[57] R. A. McDonald and P. M. Schultheiss. Information Rates of Gaussian Signals under Criteria Constraining the Error Spectrum. *Proceedings of the IEEE*, 52:415–416, 1964.

[58] A. Moffat, R. M. Neil, and I. H. Witten. Arithmetic Coding Revisited. *ACM Transactions on Information Systems*, 16(3):256–294, July 1998.

[59] P. F. Panter and W. Dite. Quantization Distortion in Pulse Code Modulation with Nonuniform Spacing of Levels. *Proc. IRE*, 39:44–48, Jan. 1951.

[60] A. Papoulis and S. U. Pillai. *Probability, Random Variables and Stochastic Processes*. McGraw-Hill, New York, NY, USA, 2002.

[61] R. Pasco. Source Coding Algorithms for Fast Data Compression. Ph.D. dissertation, Stanford University, 1976.

[62] R. L. D. Queiroz and T. D. Tran. Lapped Transforms for Image Compression. In *The Transform and Data Compression Handbook*, pages 197–265, Boca Raton, FL, 2001. CRC.

[63] J. Rissanen. Generalized Kraft Inequality and Arithmetic Coding. *IBM J. Res. Develop.*, 20:198–203, 1976.

[64] J. Rosenberg, H. Schulzrinne, G. Camarillo, A. Johnston, J. Peterson, R. Sparks, M. Handley, and E. Schooler. SIP: Session Initiation Protocol. IETF RFC 3261, 2002.

[65] A. Said. Arithmetic Coding. In K. Sayood, editor, *Lossless Compression Handbook*. Academic Press, San Diego, CA, 2003.

[66] S. A. Savari and R. G. Gallager. Generalized Tunstall Codes for Soures with Memory. *IEEE Transactions on Information Theory*, 43(2):658–668, Mar. 1997.

[67] K. Sayood, editor. *Lossless Compression Handbook*. Academic Press, San Diego, CA, 2003.

[68] H. Schulzrinne, S. Casner, R. Frederick, and V. Jacobson. RTP: A Transport Protocol for Real-Time Applications. IETF RFC 1889, 1996.

[69] C. E. Shannon. A Mathematical Theory of Communication. *The Bell System Technical Journal*, 27(3):2163–2177, July 1948.

[70] C. E. Shannon. Coding Theorems for a Discrete Source with a Fidelity Criterion. *IRE National Convention Record, Part 4*, pages 142–163, 1959.

[71] Y. Shoham and A. Gersho. Efficient Bit Allocation for an Arbitrary Set of Quantizers. *IEEE Transactions on Acoustics, Speech and Signal Processing*, 36:1445–1453, Sept. 1988.

[72] D. S. Taubman and M. M. Marcellin. *JPEG2000: Image Compression Fundamentals, Standards and Practice*. Kluwer Academic Publishers, 2001.

[73] B. P. Tunstall. Synthesis of Noiseless Compression Codes. Ph.D. dissertation, Georgia Inst. Technol., 1967.

[74] B. E. Usevitch. A Tutorial on Mondern Lossy Wavelet Image Compression: Foundations of JPEG 2000. *IEEE Signal Processing Magazine*, 18(5):22–35, Sept. 2001.

[75] P. P. Vaidyanathan. *The Theory of Linear Prediction*. Morgan & Claypool Publishers, 2008.

[76] M. Vetterli. Wavelets, Approximation, and Compression. *IEEE Signal Processing Magazine*, 18(5):59–73, Sept. 2001.

[77] M. Vetterli and J. Kovacevic. *Wavelets and Subband Coding*. Prentice-Hall, Englewood Cliffs, NJ, 1995.

[78] I. H. Witten, R. M. Neal, and J. G. Cleary. Arithmetic Coding for Data Compression. *Communications of the ACM*, 30(6):520–540, June 1987.

[79] J. Ziv and A. Lempel. A Universal Algorithm for Data Compression. *IEEE Transactions on Information Theory*, 23(3):337–343, May 1977.