Tools for Signal Compression

Tools for Signal Compression

Nicolas Moreau

First published 2011 in Great Britain and the United States by ISTE Ltd and John Wiley & Sons, Inc.
Adapted and updated from *Outils pour la compression des signaux: applications aux signaux audioechnologies du stockage d'énergie* published 2009 in France by Hermes Science/Lavoisier
© Institut Télécom et LAVOISIER 2009

ISTE Ltd	John Wiley & Sons, Inc.
27-37 St George's Road	111 River Street
London SW19 4EU	Hoboken, NJ 07030
UK	USA
www.iste.co.uk	www.wiley.com

Library of Congress Cataloging-in-Publication Data

Moreau, Nicolas, 1945-
[Outils pour la compression des signaux. English]
Tools for signal compression / Nicolas Moreau.
p. cm.
"Adapted and updated from Outils pour la compression des signaux : applications aux signaux audioechnologies du stockage d'energie."
Includes bibliographical references and index.
ISBN 978-1-84821-255-8
1. Sound--Recording and reproducing--Digital techniques. 2. Data compression (Telecommunication) 3. Speech processing systems. I. Title.
TK7881.4.M6413 2011
621.389'3--dc22

2011003206

British Library Cataloguing-in-Publication Data
A CIP record for this book is available from the British Library
ISBN 978-1-84821-255-8

Printed and bound in Great Britain by CPI Antony Rowe, Chippenham and Eastbourne.

FSC
www.fsc.org
MIX
Paper from responsible sources
FSC® C013604

Table of Contents

Introduction

In everyday life, we often come in contact with compressed signals: when using mobile telephones, mp3 players, digital cameras, or DVD players. The signals in each of these applications, telephone-band speech, high fidelity audio signal, and still or video images are not only sampled and quantized to put them into a form suitable for saving in mass storage devices or to send them across networks, but also compressed. The first operation is very basic and is presented in all courses and introductory books on signal processing. The second operation is more specific and is the subject of this book: here, the standard tools for signal compression are presented, followed by examples of how these tools are applied in compressing speech and musical audio signals. In the first part of this book, we focus on a problem which is theoretical in nature: minimizing the mean squared error. The second part is more concrete and qualifies the previous steps in seeking to minimize the bit rate while respecting the psychoacoustic constraints. We will see that signal compression consists of seeking not only to eliminate all redundant parts of the original signal but also to attempt the elimination of inaudible parts of the signal.

The compression techniques presented in this book are not new. They are explained in theoretical framework, information theory, and source coding, aiming to formalize the first (and the last) element in a digital communication channel: the encoding of an analog signal (with continuous times and continuous values) to a digital signal (at discrete times and discrete values). The techniques come from the work by C. Shannon, published at the beginning of the 1950s. However, except for the development of speech encodings in the 1970s to promote an entirely digitally switched telephone network, these techniques really came into use toward the end of the 1980s under the influence of working groups, for example, "Group Special Mobile (GSM)", "Joint Photographic Experts Group (JPEG)", and "Moving Picture Experts Group (MPEG)".

The results of these techniques are quite impressive and have allowed the development of the applications referred to earlier. Let us consider the example of a music signal. We know that a music signal can be reconstructed with quasi-perfect quality (CD quality) if it was sampled at a frequency of 44.1 kHz and quantized at a resolution of 16 bits. When transferred across a network, the required bit rate for a mono channel is 705 kb/s. The most successful audio encoding, MPEG-4 AAC, ensures “transparency” at a bit rate of the order of 64 kb/s, giving a compression rate greater than 10, and the completely new encoding MPEG-4 HE-AACv2, standardized in 2004, provides a very acceptable quality (for video on mobile phones) at 24 kb/s for 2 stereo channels. The compression rate is better than 50!

In the Part 1 of this book, the standard tools (scalar quantization, predictive quantization, vector quantization, transform and sub-band coding, and entropy coding) are presented. To compare the performance of these tools, we use an academic example of the quantization of the realization $x(n)$ of a one-dimensional random process $X(n)$. Although this is a theoretical approach, it not only allows objective assessment of performance but also shows the coherence between all the available tools. In the Part 2, we concentrate on the compression of audio signals (telephone-band speech, wideband speech, and high fidelity audio signals).

Throughout this book, we discuss the basic ideas of signal processing using the following language and notation. We consider a one-dimensional, stationary, zero-mean, random process $X(n)$, with power σ_X^2 and power spectral density $S_X(f)$. We also assume that it is Gaussian, primarily because the Gaussian distribution is preserved in all linear transformations, especially in a filter which greatly simplifies the notation, and also because a Gaussian signal is the most difficult signal to encode because it carries the greatest quantization error for any bit rate. A column vector of N dimensions is denoted by $\underline{X}(m)$ and constructed with $X(mN) \cdots X(mN+N-1)$. These N random variables are completely defined statistically by their probability density function:

$$p_X(\underline{x}) = \frac{1}{(2\pi)^{N/2}\sqrt{\det \mathbb{R}_X}} \exp(-\frac{1}{2}\underline{x}^t \mathbb{R}_X^{-1} \underline{x})$$

where $\mathbb{R}_X$ is the autocovariance matrix:

$$\mathbb{R}_X = \mathbb{E}\{\underline{X}(m)\underline{X}^t(m)\} = \begin{bmatrix} r_X(0) & r_X(1) & \cdots & r_X(N-1) \\ r_X(1) & \ddots & \ddots & \vdots \\ \vdots & \ddots & \ddots & r_X(1) \\ r_X(N-1) & \cdots & r_X(1) & r_X(0) \end{bmatrix}$$

Toeplitz matrix with $N \times N$ dimensions. Moreover, we assume an auto-regressive process $X(n)$ of order P, obtained through filtering with white noise $W(n)$ with variance σ_W^2 via a filter of order P with a transfer function $1/A(z)$ for $A(z)$ in the form:

$$A(z) = 1 + a_1 z^{-1} + \cdots + a_P z^{-P}$$

The purpose of considering the quantization of an auto-regressive waveform as our example is that it allows the simple explanation of all the statistical characteristics of the source waveform as a function of the parameters of the filter such as, for example, the power spectral density:

$$S_X(f) = \frac{\sigma_W^2}{|A(f)|^2}$$

where the notation $A(f)$ is inaccurate and should be more properly written as $A(\exp(j2\pi f))$. It also allows us to give analytical expressions for the quantization error power for different quantization methods when quadratic error is chosen as the measure of distortion. Comparison of the performance of the different methods is thereby possible. From a practical point of view, this example is not useless because it is a reasonable model for a number of signals, for example, for speech signals (which are only locally stationary) when the order P selected is high enough (e.g. 8 or 10).

Part 1

Tools for Signal Compression

Chapter 1

Scalar Quantization

1.1. Introduction

Let us consider a discrete-time signal $x(n)$ with values in the range $[-A, +A]$. Defining a scalar quantization with a resolution of b bits per sample requires three operations:

– partitioning the range $[-A, +A]$ into $L = 2^b$ non-overlapping intervals $\{\Theta^1 \cdots \Theta^L\}$ of length $\{\Delta^1 \cdots \Delta^L\}$,

– numbering the partitioned intervals $\{i^1 \cdots i^L\}$,

– selecting the reproduction value for each interval, the set of these reproduction values forms a dictionary (codebook)[1] $C = \{\hat{x}^1 \cdots \hat{x}^L\}$.

Encoding (in the transmitter) consists of deciding which interval $x(n)$ belongs to and then associating it with the corresponding number $i(n) \in \{1 \cdots L = 2^b\}$. It is the number of the chosen interval, the symbol, which is transmitted or stored. The decoding procedure (at the receiver) involves associating the corresponding reproduction value $\hat{x}(n) = \hat{x}^{i(n)}$ from the set of reproduction values $\{\hat{x}^1 \cdots \hat{x}^L\}$ with the number $i(n)$. More formally, we observe that quantization is a non-bijective mapping to $[-A, +A]$ in a finite set C with an assignment rule:

$$\hat{x}(n) = \hat{x}^{i(n)} \in \{\hat{x}^1 \cdots \hat{x}^L\} \quad \text{iff} \quad x(n) \in \Theta^i$$

The process is irreversible and involves loss of information, a quantization error which is defined as $q(n) = x(n) - \hat{x}(n)$. The definition of a distortion measure

1. In scalar quantization, we usually speak about quantization levels, quantization steps, and decision thresholds. This language is also adopted for vector quantization.

$d[x(n), \hat{x}(n)]$ is required. We use the simplest distortion measure, quadratic error:

$$d[x(n), \hat{x}(n)] = |x(n) - \hat{x}(n)|^2$$

This measures the error in each sample. For a more global distortion measure, we use the mean squared error (MSE):

$$D = \mathbb{E}\{|X(n) - \hat{x}(n)|^2\}$$

This error is simply denoted as the quantization error power. We use the notation σ_Q^2 for the MSE.

Figure 1.1(a) shows, on the left, the signal before quantization and the partition of the range $[-A, +A]$ where $b = 3$, and Figure 1.1(b) shows the reproduction values, the reconstructed signal and the quantization error. The bitstream between the transmitter and the receiver is not shown.

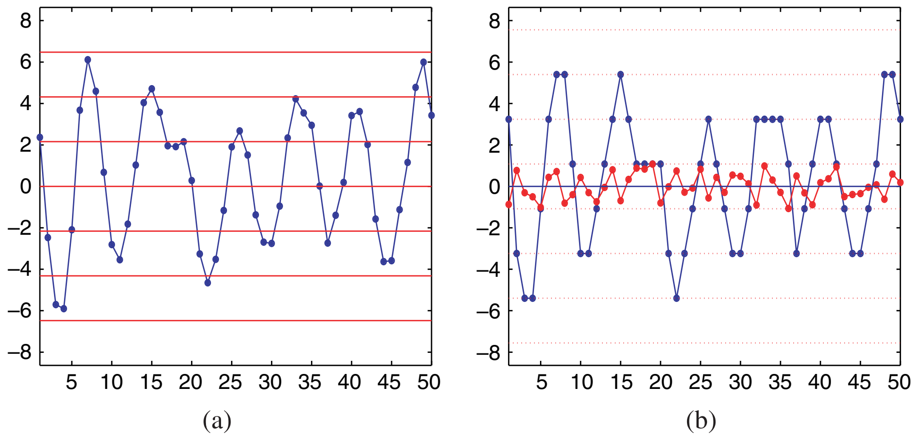

Figure 1.1. *(a) The signal before quantization and the partition of the range $[-A, +A]$ and (b) the set of reproduction values, reconstructed signal, and quantization error*

The problem now consists of defining the optimal quantization, that is, in defining the intervals $\{\Theta^1 \cdots \Theta^L\}$ and the set of reproduction values $\{\hat{x}^1 \cdots \hat{x}^L\}$ to minimize σ_Q^2.

1.2. Optimum scalar quantization

Assume that $x(n)$ is the realization of a real-valued stationary random process $X(n)$. In scalar quantization, what matters is the distribution of values that the random process $X(n)$ takes at time n. No other direct use of the correlation that exists between the values of the process at different times is possible. It is enough to know the marginal probability density function of $X(n)$, which is written as $p_X(.)$.

1.2.1. *Necessary conditions for optimization*

To characterize the optimum scalar quantization, the range partition and reproduction values must be found which minimize:

$$\sigma_Q^2 = \mathbb{E}\{[X(n) - \hat{x}(n)]^2\} = \sum_{i=1}^{L} \int_{u \in \Theta^i} (u - \hat{x}^i)^2 p_X(u) \mathrm{d}u \qquad [1.1]$$

This joint minimization is not simple to solve. However, the two necessary conditions for optimization are straightforward to find. If the reproduction values $\{\hat{x}^1 \cdots \hat{x}^L\}$ are known, the best partition $\{\Theta^1 \cdots \Theta^L\}$ can be calculated. Once the partition is found, the best reproduction values can be deduced. The encoding part of quantization must be optimal if the decoding part is given, and vice versa. These two necessary conditions for optimization are simple to find when the squared error is chosen as the measure of distortion.

– Condition 1: Given a codebook $\{\hat{x}^1 \cdots \hat{x}^L\}$, the best partition will satisfy:

$$\Theta^i = \{x : (x - \hat{x}^i)^2 \leq (x - \hat{x}^j)^2 \quad \forall j \in \{1 \cdots L\}\ \}$$

This is the nearest neighbor rule.

If we define t^i such that it defines the boundary between the intervals Θ^i and Θ^{i+1}, minimizing the MSE σ_Q^2 relative to t^i is found by noting:

$$\frac{\partial}{\partial t^i} \left[\int_{t^{i-1}}^{t^i} (u - \hat{x}^i)^2 p_X(u) \mathrm{d}u + \int_{t^i}^{t^{i+1}} (u - \hat{x}^{i+1})^2 p_X(u) \mathrm{d}u \right] = 0$$

$$(t^i - \hat{x}^i)^2 p_X(t^i) - (t^i - \hat{x}^{i+1})^2 p_X(t^i) = 0$$

such that:

$$t^i = \frac{\hat{x}^i + \hat{x}^{i+1}}{2}$$

– Condition 2: Given a partition $\{\Theta^1 \cdots \Theta^L\}$, the optimum reproduction values are found from the centroid (or center of gravity) of the section of the probability density function in the region of Θ^i:

$$\hat{x}^i = \frac{\int_{u \in \Theta^i} u p_X(u) \mathrm{d}u}{\int_{u \in \Theta^i} p_X(u) \mathrm{d}u} = \mathbb{E}\{X | X \in \Theta^i\} \qquad [1.2]$$

First, note that minimizing σ_Q^2 relative to $\hat{x}^i$ involves only an element from the sum given in [1.1]. From the following:

$$\frac{\partial}{\partial \hat{x}^i} \int_{u \in \Theta^i} (u - \hat{x}^i)^2 p_X(u) \mathrm{d}u = 0$$

$$-2 \int_{u \in \Theta^i} u p_X(u) \mathrm{d}u + 2\hat{x}^i \int_{u \in \Theta^i} p_X(u) \mathrm{d}u = 0$$

we find the first identity of equation [1.2].

Since:

$$\int_{u \in \Theta^i} u p_X(u) \mathrm{d}u = \int_{u \in \Theta^i} p_X(u) \mathrm{d}u \int_{-\infty}^{\infty} u p_{X|\Theta^i}(u) \mathrm{d}u$$

where $p_{X|\Theta^i}$ is the conditional probability density function of X, where $X \in \Theta^i$, we find:

$$\hat{x}^i = \int_{-\infty}^{\infty} u p_{X|\Theta^i}(u) \mathrm{d}u$$

$$\hat{x}^i = \mathbb{E}\{X | X \in \Theta^i\}$$

The required value is the mean value of X in the interval under consideration.[2]

It can be demonstrated that these two optimization conditions are not sufficient to guarantee optimized quantization except in the case of a Gaussian distribution.

Note that detailed knowledge of the partition is not necessary. The partition is determined entirely by knowing the distortion measure, applying the nearest neighbor rule, and from the set of reproduction values. Figure 1.2 shows a diagram of the encoder and decoder.

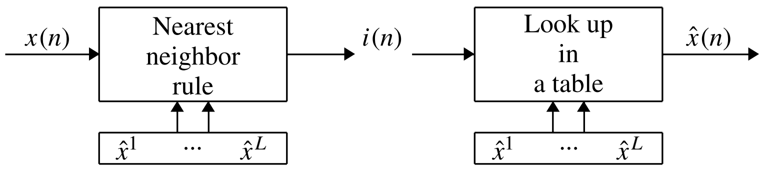

Figure 1.2. *Encoder and decoder*

2. This result can be interpreted in a mechanical system: the moment of inertia of an object with respect to a point is at a minimum when the point is the center of gravity.

1.2.2. *Quantization error power*

When the number L of levels of quantization is high, the optimum partition and the quantization error power can be obtained as a function of the probability density function $p_X(x)$, unlike in the previous case. This hypothesis, referred to as the *high-resolution* hypothesis, declares that the probability density function can be assumed to be constant in the interval $[t^{i-1}, t^i]$ and that the reproduction value is located at the middle of the interval. We can therefore write:

$$p_X(x) \approx p_X(\hat{x}^i) \quad \text{for } \; x \in [t^{i-1}, t^i]$$

$$\hat{x}^i \approx \frac{t^{i-1} + t^i}{2}$$

We define the length of the interval as:

$$\Delta(i) = t^i - t^{i-1}$$

for an interval $[t^{i-1}, t^i]$ and:

$$P_{\text{prob}}(i) = P_{\text{prob}}\{X \in [t^{i-1}, t^i]\} = p_X(\hat{x}^i)\Delta(i)$$

is the probability that $X(n)$ belongs to the interval $[t^{i-1}, t^i]$. The quantization error power is written as:

$$\sigma_Q^2 = \sum_{i=1}^{L} p_X(\hat{x}^i) \int_{t^{i-1}}^{t^i} (u - \hat{x}^i)^2 \mathrm{d}u$$

Since:

$$\int_{t^{i-1}}^{t^i} (u - \hat{x}^i)^2 \mathrm{d}u = \int_{-\Delta(i)/2}^{+\Delta(i)/2} u^2 \mathrm{d}u = \frac{\Delta^3(i)}{12}$$

we find:

$$\sigma_Q^2 = \frac{1}{12} \sum_{i=1}^{L} p_X(\hat{x}^i)\Delta^3(i) \qquad [1.3]$$

This is also written as:

$$\sigma_Q^2 = \sum_{i=1}^{L} P_{\text{prob}}(i) \frac{\Delta^2(i)}{12} = \mathbb{E}\left\{\frac{\Delta^2}{12}\right\}$$

The quantization error power depends only on the length of the intervals $\Delta(i)$. We are looking for $\{\Delta(1) \cdots \Delta(L)\}$ such that σ_Q^2 is minimized. Let:

$$\alpha^3(i) = p_X(\hat{x}^i)\Delta^3(i)$$

As:

$$\sum_{i=1}^{L} \alpha(i) = \sum_{i=1}^{L} [p_X(\hat{x}^i)]^{1/3}\Delta(i) \approx \int_{-\infty}^{+\infty} [p_X(u)]^{1/3}\mathrm{d}u = const$$

since this integral is now independent of $\Delta(i)$, we minimize the sum of the cubes of L positive numbers with a constant sum. This is satisfied with numbers that are all equal. Hence, we have:

$$\alpha(1) = \cdots = \alpha(L)$$

which implies:

$$\alpha^3(1) = \cdots = \alpha^3(L)$$
$$p_X(\hat{x}^1)\Delta^3(1) = \cdots = p_X(\hat{x}^L)\Delta^3(L)$$

The relation means that an interval is even smaller, that the probability that $X(n)$ belongs to this interval is high, and that all the intervals contribute equally to the quantization error power. The expression for the quantization error power is:

$$\sigma_Q^2 = \frac{L}{12}\alpha^3$$

where:

$$\alpha = \frac{1}{L}\int_{-\infty}^{+\infty} [p_X(u)]^{1/3}\mathrm{d}u$$

Hence, we have:

$$\sigma_Q^2 = \frac{1}{12L^2}\left(\int_{-\infty}^{+\infty} [p_X(u)]^{1/3}\mathrm{d}u\right)^3$$

Since $L = 2^b$, we obtain what is known as the Bennett formula:

$$\sigma_Q^2 = \frac{1}{12}\left(\int_{-\infty}^{+\infty} [p_X(u)]^{1/3}\mathrm{d}u\right)^3 2^{-2b} \qquad [1.4]$$

This demonstration is not mathematically rigorous. It will be discussed at the end of Chapter 4 where we compare this mode of quantization with what is known as quantization with entropy constraint.

Two cases are particularly interesting. When $X(n)$ is distributed uniformly, we find:

$$\sigma_Q^2 = \frac{A^2}{3}\, 2^{-2b} = \sigma_X^2\, 2^{-2b}$$

Note that the explanation via Bennett's formula is not necessary. We can obtain this result directly!

For a Gaussian zero-mean signal, with power σ_X^2, for which:

$$p_X(x) = \frac{1}{\sqrt{2\pi\sigma_X^2}} \exp\left(-\frac{x^2}{2\sigma_X^2}\right)$$

we have:

$$\int_{-\infty}^{+\infty} [p_X(u)]^{1/3} \mathrm{d}u = \int_{-\infty}^{+\infty} \frac{1}{(2\pi\sigma_X^2)^{1/6}} \exp\left(-\frac{x^2}{6\sigma_X^2}\right) \mathrm{d}u$$

$$\int_{-\infty}^{+\infty} [p_X(u)]^{1/3} \mathrm{d}u = (2\pi\sigma_X^2)^{1/3}\sqrt{3} \int_{-\infty}^{+\infty} \frac{1}{(2\pi 3\sigma_X^2)^{1/2}} \exp\left(-\frac{x^2}{6\sigma_X^2}\right) \mathrm{d}u$$

$$\int_{-\infty}^{+\infty} [p_X(u)]^{1/3} \mathrm{d}u = (2\pi\sigma_X^2)^{1/3}\sqrt{3}$$

From this, we deduce that:

$$\sigma_Q^2 = \frac{1}{12} 2\pi\sigma_X^2 \, 3^{3/2} \, 2^{-2b}$$

$$\sigma_Q^2 = c\, \sigma_X^2 \, 2^{-2b} \qquad [1.5]$$

where:

$$c = \frac{\sqrt{3}}{2}\pi$$

This equation is referred to throughout this book. From this, we can write the equivalent expression:

$$10 \log_{10} \frac{\sigma_X^2}{\sigma_Q^2} = 6.05b - 4.35 \text{ dB}$$

From this we deduce the *6 dB per bit rule*. We can show that for all other distributions (Laplacian, etc.), the minimum quantization error power is always between these two values. The case of the uniformly distributed signal is more favorable, whereas the Gaussian case is less favorable. Shannon's work and the rate/distortion theory affirm this observation.

It is interesting to know the statistical properties of the quantization error. We can show that the quantization error is not correlated to the reconstructed signal but this property is not true for the original signal. We can also show that, only in the framework of the high-resolution hypothesis, the quantization error can be modeled by white noise. A detailed analysis is possible (see [LIP 92]).

1.2.3. *Further information*

1.2.3.1. *Lloyd–Max algorithm*

In practice, $p_X(x)$ is unknown. To construct a quantizer, we use empirical data, assign the same weight to each value and apply the Lloyd–Max algorithm in the so-called Linde-Buzo-Gray (LBG) form. This algorithm, which is generalized for vector quantization, is presented in the following chapter.

1.2.3.2. *Non-linear transformation*

A non-uniform scalar quantizer can be seen as a uniform scalar quantizer that has been preceded by a nonlinear transformation and followed with the inverse transformation.[3] The transformation is defined by its characteristic $f(x)$. From this perspective, the problem consists of choosing the non-linear transformation which minimizes the quantization error power. This forms the subject of important developments in two works by: Jayant and Noll [JAY 84] and Gersho and Gray [GER 92]. This development no longer seems to be of great importance because vector quantization became the basic tool of choice.

1.2.3.3. *Scale factor*

During a quantization operation on real signals (speech, music, and pictures), it is important to estimate the parameter A which varies with time; real signals do not satisfy the stationarity hypothesis! We examine this problem in the following chapter by introducing a special quantization called *gain shape* which is particularly well adapted to signals with significant instantaneous changes in power, for example, audio signals.

1.3. Predictive scalar quantization

1.3.1. *Principle*

In the preceding exposition, we saw that during quantization, no other use is made with the statistical links between successive values of the signal. We will see that predictive scalar quantization aims to decorrelate the signal before quantizing it and that the use of correlation improves the general behavior of the system, that is, it reduces the quantization error power.

An outline of the principle of predictive scalar quantization is shown in Figure 1.3. We subtract a new signal $v(n)$ from the signal $x(n)$. Next, we perform the encoding/decoding procedure on the signal $y(n) = x(n) - v(n)$. At the decoder, we add $v(n)$ back to the reconstructed signal values $\hat{y}(n)$.

3. We use the neologism companding for *compressing* + *expanding*.

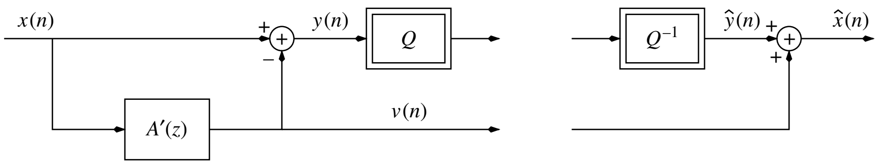

Figure 1.3. *Outline of the principle of predictive scalar quantization*

We can immediately see that, in a real-world application of coding, this scheme is not very realistic since the signal $v(n)$ must also be transmitted to the decoder, but let us wait until the end of the chapter before demonstrating how we go from a *open-loop* scheme to a more realistic, but more complicated to analyze, *closed-loop* scheme.

If we subtract a value from the signal before encoding it and add it back after decoding, the quantization error $q(n) = y(n) - \hat{y}(n)$ and reconstruction error $\bar{q}(n) = x(n) - \hat{x}(n)$ must always be equal because:

$$q(n) = y(n) - \hat{y}(n) = x(n) - v(n) - [\hat{x}(n) - v(n)] = \bar{q}(n)$$

Hence their respective powers are identical. Since the main interest of the user of the complete system is to have the smallest possible reconstruction error power, the problem becomes simply the reduction of the quantization error power. If we assume an optimized scalar quantization of $y(n)$, we know that the quantization error power can be expressed as:

$$\sigma_Q^2 = c\, \sigma_Y^2\, 2^{-2b}$$

From this, we conclude that seeking to minimize the reconstruction error power σ_Q^2 leads us back to minimize σ_Y^2 from $y(n)$.

We have a great range of choices for $v(n)$. If we take $v(n)$ in the form:

$$v(n) = -\sum_{i=1}^{P} a_i x(n-i)$$

while introducing P parameters, we speak of *linear prediction of order* P. The signal $y(n)$ is the prediction error which is expressed as:

$$y(n) = x(n) - v(n) = x(n) + \sum_{i=1}^{P} a_i x(n-i)$$

The relationship between $x(n)$ and $y(n)$ is that of a transfer function filtering operation[4]:

$$B(z) = 1 + a_1 z^{-1} + \cdots + a_P z^{-P}$$

Minimizing σ_Y^2 concerns the coefficients of this predictive filter.

This problem has been the subject of numerous studies since 1960s. All modern books that present basic techniques for signal processing include a chapter on this problem, for example [KAY 88]. In this book, we set out a few reminders.

1.3.2. *Reminders on the theory of linear prediction*

1.3.2.1. *Introduction: least squares minimization*

Since this theory, which can be used in numerous signal processing applications, was developed quite rightly with the goal of determining a method of speech coding with reduced rates and, in coding speech, the method uses a block of N samples,[5] the problem can be posed in the following way: knowing $\underline{x} = [x(0) \cdots x(N-1)]^t$ determine $\underline{a} = [a_1 \cdots a_P]^t$ while minimizing the empirical power of the prediction error:

$$\underline{a}^{\text{opt}} = \arg\min_{\underline{a}} \hat{\sigma}_Y^2$$

where:

$$\hat{\sigma}_Y^2 = \frac{1}{N} \sum_{n=0}^{N-1} y^2(n) = \frac{1}{N} \underline{y}^t \, \underline{y}$$

Since:

$$\begin{bmatrix} y(0) \\ y(1) \\ \vdots \\ y(N-1) \end{bmatrix} = \begin{bmatrix} x(0) \\ x(1) \\ \vdots \\ x(N-1) \end{bmatrix} + \begin{bmatrix} x(-1) & \cdots & x(-P) \\ x(0) & \cdots & x(-P+1) \\ \vdots & \ddots & \vdots \\ x(N-2) & \cdots & x(N-P-1) \end{bmatrix} \begin{bmatrix} a_1 \\ \vdots \\ a_P \end{bmatrix}$$

such that:

$$\underline{y} = \underline{x} + \Gamma \underline{a}$$

4. Despite using the same notation to avoid overwhelming the reader, we must not confuse the coefficients a_i and the order P of the generating filter of $x(n)$ with the coefficients and the order of the predictor polynomial. Throughout this chapter, we are only concerned with the predictor filter.

5. We will discuss in the second half of this book.

we can write:

$$\hat{\sigma}_Y^2 = \frac{1}{N}(\underline{x} + \Gamma\underline{a})^t(\underline{x} + \Gamma\underline{a}) = \frac{1}{N}(\underline{x}^t\underline{x} + 2(\Gamma^t\underline{x})^t\underline{a} + \underline{a}^t\Gamma^t\Gamma\underline{a})$$

The vector $\underline{a}^{\rm opt}$ is that which satisfies:

$$\frac{\partial}{\partial \underline{a}}\hat{\sigma}_Y^2 = 2\Gamma^t\underline{x} + 2\Gamma^t\Gamma\underline{a}^{\rm opt} = \underline{0}$$

If $\Gamma^t\Gamma$ is invertible, we find:

$$\underline{a}^{\rm opt} = -(\Gamma^t\Gamma)^{-1}\Gamma^t\underline{x}$$

The minimum can be written as:

$$\hat{\sigma}_Y^2 = \frac{1}{N}[\underline{x}^t\underline{x} + 2(\Gamma^t\underline{x})^t\underline{a}^{\rm opt} + (\underline{a}^{\rm opt})^t(-\Gamma^t\underline{x})] = \frac{1}{N}\underline{x}^t\underline{x} + \frac{1}{N}(\Gamma^t\underline{x})^t\underline{a}^{\rm opt}$$

1.3.2.2. *Theoretical approach*

Let us assume that the signal $x(n)$ can be interpreted (modeled) as the realization of a stationary random process with an autocovariance function $r_X(k) = E\{X(n)X(n-k)\}$. We need to find the vector $\underline{a}$ which minimizes the prediction error power:

$$\sigma_Y^2 = \mathbb{E}\{Y^2(n)\} = \mathbb{E}\{[X(n) + \sum_{i=1}^{P} a_i X(n-i)]^2\}$$

$$\sigma_Y^2 = \mathbb{E}\{X^2(n)\} + 2\sum_{i=1}^{P} a_i \mathbb{E}\{X(n)X(n-i)\} + \sum_{i=1}^{P}\sum_{j=1}^{P} a_i a_j \mathbb{E}\{X(n-i)X(n-j)\}$$

$$\sigma_Y^2 = \sigma_X^2 + 2\underline{a}\begin{bmatrix} r_X(1) \\ \vdots \\ r_X(P)\end{bmatrix} + \underline{a}^t \begin{bmatrix} r_X(0) & \cdots & r_X(P-1) \\ \vdots & \ddots & \vdots \\ r_X(P-1) & \cdots & r_X(0)\end{bmatrix}\underline{a}$$

$$\sigma_Y^2 = \sigma_X^2 + 2\underline{r}^t\underline{a} + \underline{a}^t\mathbb{R}\underline{a}$$

Minimizing σ_Y^2 relative to $\underline{a}$ involves the *normal equations*:

$$\mathbb{R}\underline{a}^{\rm opt} = -\underline{r}$$

and as the autocovariance matrix $\mathbb{R}$ is definite-positive (except for the limiting case where $X(n)$ is an harmonic random process), it is invertible. We therefore have:

$$\underline{a}^{\rm opt} = -\mathbb{R}^{-1}\underline{r} \qquad [1.6]$$

We also have:

$$(\sigma_Y^2)^{\min} = \sigma_X^2 + 2(\underline{a}^{\text{opt}})^t \underline{r} - (\underline{a}^{\text{opt}})^t \underline{r} = \sigma_X^2 + (\underline{a}^{\text{opt}})^t \underline{r} \qquad [1.7]$$

Note that these two equations together [1.6] and [1.7] allow the unique matrix representation:

$$\begin{bmatrix} r_X(0) & r_X(1) & \cdots & r_X(P) \\ r_X(1) & \ddots & \ddots & \vdots \\ \vdots & \ddots & \ddots & r_X(1) \\ r_X(P) & \cdots & r_X(1) & r_X(0) \end{bmatrix} \begin{bmatrix} 1 \\ a_1 \\ \vdots \\ a_P \end{bmatrix} = \begin{bmatrix} \sigma_Y^2 \\ 0 \\ \vdots \\ 0 \end{bmatrix} \qquad [1.8]$$

1.3.2.3. *Comparing the two approaches*

The two solutions are comparable, which is unsurprising because $1/N\Gamma^t\underline{x}$ is an estimate of the vector $\underline{r}$ and $1/N\Gamma^t\Gamma$ is an estimate of $\mathbb{R}$. More precisely, the two approaches are asymptotically equivalent since, when the signal $X(n)$ is an ergodic random process, that is, if:

$$\lim_{N\to\infty} \frac{1}{2N+1} \sum_{n=-N}^{+N} x(n)x(n+k) = \mathbb{E}\{X(n)X(n+k)\}$$

we have:

$$\lim_{N\to\infty} \frac{\Gamma^t\Gamma}{N} = \mathbb{R} \quad \text{and} \quad \lim_{N\to\infty} \frac{\Gamma^t\underline{x}}{N} = \underline{r}$$

The difference (essential but subtle) is that since the exact covariance matrix is definite-positive (except in the limiting case where the process is harmonic), it is always invertible while the matrix $\Gamma^t\Gamma$ (expressed here with $P = 1$ for simplicity):

$$\begin{bmatrix} x^2(1) + \cdots x^2(N-2) & x(0)x(1) + \cdots + x(N-3)x(N-2) \\ x(0)x(1) + \cdots + x(N-3)x(N-2) & x^2(0) + \cdots x^2(N-3) \end{bmatrix}$$

stays symmetric but is not always definite-positive.

In practice when we have only N observed data, we wish to maintain the positive property of the matrix. We can see that to approximate the autocovariance function as:

$$\hat{r}_X(k) = \frac{1}{N} \sum_{n=0}^{N-1-k} x(n)x(n+k) \quad \text{for } k = 0 \cdots P$$

then to construct $\hat{\mathbb{R}}$ and $\hat{\underline{r}}$ from $\hat{r}_X(k)$ maintains the definite-positive characteristic of $\hat{\mathbb{R}}$ which therefore allows its invertibility. We can then define the filter coefficients by:

$$\underline{a}^{\text{opt}} = -\hat{\mathbb{R}}^{-1}\hat{\underline{r}} \qquad [1.9]$$

and the residual signal power by:

$$\hat{\sigma}_Y^2 = \hat{r}_X(0) + (\underline{a}^{\text{opt}})^t \underline{\hat{r}}$$

This is called *linear predictive coding* (LPC).

We can also show that the positive property requires that all the zero values of polynomial $A(z)$ are inside the unit circle which assures the stability of the filter $1/A(z)$. This is a very important property in practice as we will see shortly when we look at coding speech at a reduced rate.

1.3.2.4. *Whitening filter*

We can show that the prediction error $Y(n)$ is white (more precisely, the signal $X(n)$ is whitened). Note that:

$$\frac{\partial E\{|Y(n)|^2\}}{\partial a_i} = 0 \quad \Rightarrow \quad E\{Y(n)X(n-i)\} = 0 \quad \forall i = 1 \cdots P$$

Assume that P is large. As $Y(n)$ is not correlated with the preceding $X(n-i)$ and $Y(n-i)$ is a linear combination of the $X(n-i)$, we can deduce from this that $Y(n)$ is not correlated with $Y(n-i)$. The prediction error $Y(n)$ is therefore white noise but this property is not proven *a priori* unless $P \to \infty$ (asymptotic behavior). This filter, which gives $Y(n)$ from $X(n)$, is called the "whitening filter."

If $Y(n)$ is completely whitened, we can write:

$$S_Y(f) = |A(f)|^2 S_X(f) = \sigma_Y^2$$

Hence, we have:

$$S_X(f) = \frac{\sigma_Y^2}{|A(f)|^2}$$

Recall that the most regularly used spectral density estimate is the periodogram. From N observed data $[x(0) \ \cdots \ x(N-1)]$, we derive:

$$S_X\left(f_k = \frac{k}{N}\right) = \frac{1}{N} | \sum_{n=0}^{N-1} x(n) \exp\left(-j2\pi \frac{k}{N} n\right) |^2 \quad \text{for } k = 0 \cdots N/2$$

The equation $S_X(f) = \sigma_Y^2 / |A(f)|^2$ hints at a second spectral density estimate. From N observed data, we calculate the whitening filter coefficients through an LPC analysis before using the preceding equation.

1.3.2.5. *Levinson algorithm*

To determine the optimum predictive filter with order P, we must solve the linear system of P equations with P unknowns given by [1.6] or [1.9]. We can use, for example, Gauss's algorithm which requires $O(P^3)$ operations. Fast algorithms exist with $O(P^2)$ operations which make use of the centro-symmetric properties of the matrix $\mathbb{R}$; these algorithms were of much interest during the 1960s because they were fast. Currently, they are still interesting because they introduce parameters which are equivalent to a_i coefficients with good coding attributes/properties.

The best-known algorithm is the *Levinson algorithm*. We provide the description of this algorithm with no other justification. Further details can be found in for example [KAY 88]. The algorithm is recursive as follows: knowing the optimum predictor with order j, we find the predictor with order $j+1$. Note that $a_1^j \cdots a_j^j$ are the coefficients for the predictor with order j, $\rho_j = r_X(j)/r_X(0)$ are the normalized coefficients of autocovariance, and σ_j^2 is the variance of the prediction error for this order. When the index j reaches the order P, we have $a_i^{j=P} = a_i$ and $\sigma_{j=P}^2 = \sigma_Y^2$.

```
– $a_1^1 = k_1 = -\rho_1$
– $\sigma_1^2 = (1 - k_1^2)\sigma_X^2$
– For $j = 2 \cdots P$
    * $\delta_j = \rho_j + a_1^{j-1}\rho_{j-1} + \cdots + a_{j-1}^{j-1}\rho_1$
    * $k_j = -\delta_j / \sigma_{j-1}^2$
    * For $i = 1 \cdots j-1 : a_i^j = a_i^{j-1} + k_j a_{j-i}^{j-1}$
    * $a_j^j = k_j$
    * $\sigma_j^2 = \sigma_{j-1}^2 + k_j\delta_j = \sigma_{j-1}^2(1 - k_j^2)$
```

Writing the set of equation [1.8] in matrix form for $j = 0 \cdots P$ without looking to make explicit the upper triangular part of the matrix to the second member, we find[6]:

$$\mathbb{R} \begin{bmatrix} 1 & 0 & \cdots & 0 \\ a_1^P & \ddots & \ddots & \vdots \\ \vdots & \ddots & \ddots & 0 \\ a_P^P & \cdots & a_1^1 & 1 \end{bmatrix} = \begin{bmatrix} \sigma_P^2 & x & \cdots & x \\ 0 & \ddots & \ddots & \vdots \\ \vdots & \ddots & \ddots & x \\ 0 & \cdots & 0 & \sigma_0^2 \end{bmatrix} \qquad [1.10]$$

which can be interpreted as a Choleski decomposition of the autocovariance matrix.

6. Note that this time the autocovariance matrix $\mathbb{R}$ has $P+1$ dimensions, whereas it had P dimensions beforehand. We do not distinguish the matrices notationally as they would become unwieldy; in any case distinguishing them is not really necessary.

The coefficients $k_1 \cdots k_P$ are known as the partial correlation coefficients (PARCOR). The final equation in the above algorithm shows that all coefficients have a magnitude less than one because the variances are always positive. This property is what makes them particularly interesting in coding.

1.3.3. *Prediction gain*

1.3.3.1. *Definition*

The prediction gain (gain which arises from prediction) is the ratio of the quantization error powers, obtained through using the optimum quantization without prediction, with those obtained when using prediction with a constant resolution b. It is equal to:

$$G_{\mathrm{p}} = \frac{c\,\sigma_X^2\, 2^{-2b}}{c\,\sigma_Y^2\, 2^{-2b}} = \frac{\sigma_X^2}{\sigma_Y^2} \qquad [1.11]$$

Making use of [1.7], the prediction gain can be written as:

$$G_{\mathrm{p}}(P) = \frac{r_X(0)}{r_X(0) + \sum_{i=1}^{P} a_i^{\mathrm{opt}} r_X(i)}$$

It depends on the prediction order P. The prediction gain can also be written as a function of the PAR-COR. Since $\sigma_Y^2 = \sigma_X^2 \prod_{i=1}^{P}(1 - k_i^2)$, the prediction gain is equal to:

$$G_{\mathrm{p}}(P) = \frac{1}{\prod_{i=1}^{P}(1 - k_i^2)}$$

This function increases with P. We can show that it tends toward the limit $G_{\mathrm{p}}(\infty)$ which is known as the asymptotic value of the prediction gain.

1.3.4. *Asymptotic value of the prediction gain*

This asymptotic value can be expressed in different ways, for example, as a function of the autocovariance matrix determinant. By taking the determinant of the two parts of equation [1.10], we find[7]:

$$\det \mathbb{R}(P+1) = \prod_{j=0}^{P} \sigma_j^2$$

7. Specifying explicitly the dimension of the matrix $\mathbb{R}$ in this section.

In general, when the prediction order is increased, the prediction error power decreases rapidly before staying practically constant from a certain order P_0. This happens because the signal is not correlated beyond this order and we cannot improve the prediction by increasing the order. With σ_Y^2 being the smallest power possible and for a sufficiently large P, we have:

$$\det \mathbb{R}(P+1) \approx (\sigma_Y^2)^{P+1}$$

Therefore, we have:

$$G_{\mathrm{p}}(\infty) = \frac{\sigma_X^2}{\lim_{P\to\infty}[\det \mathbb{R}(P)]^{1/P}} \qquad [1.12]$$

The asymptotic value of the prediction gain can also be expressed as a function of the power spectral density $S_X(f)$ of the random process $X(n)$.

First of all, let us demonstrate that the prediction error power while using all of the above for a time-invariant process with power spectral density $S_X(f)$ is given by:

$$\sigma_Y^2 = \exp\left(\int_{-1/2}^{1/2} \log_e S_X(f)\mathrm{d}f\right)$$

In effect, we write $B(z)$ in the form:

$$B(z) = 1 + \sum_{i=1}^{P} a_i^{\mathrm{opt}} z^{-i} = \prod_{i=1}^{P}(1 - p_i z^{-1}) \quad \text{with} \quad |p_i| < 1$$

because the polynomial $B(z)$ is minimum-phase. We can therefore write:

$$\log_e[\prod_{i=1}^{P}(1 - p_i z^{-1})] = \sum_{i=1}^{P} \log_e(1 - p_i z^{-1}) = \alpha_1 z^{-1} + \alpha_2 z^{-2} + \cdots$$

for z which belongs to an existence domain which includes the unit circle. We therefore have:

$$\int_{-1/2}^{+1/2} \log_e[B(f)]\mathrm{d}f = 0$$

through the application of Cauchy's theorem. From there, we deduce:

$$\int_{-1/2}^{+1/2} \log_e[S_X(f)]\mathrm{d}f = \int_{-1/2}^{+1/2} \log_e(\sigma_Y^2)\mathrm{d}f - \int_{-1/2}^{+1/2} \log_e(|B(f)|^2)\mathrm{d}f = \log_e(\sigma_Y^2)$$

which leads to the result.

Because the signal power is equal to:

$$\sigma_X^2 = \int_{-1/2}^{1/2} S_X(f)\mathrm{d}f$$

we find a new expression for the asymptotic value of the prediction gain:

$$G_\mathrm{p}(\infty) = \frac{\int_{-1/2}^{1/2} S_X(f)\mathrm{d}f}{\exp(\int_{-1/2}^{1/2} \log_e S_X(f)\mathrm{d}f)} \qquad [1.13]$$

as a function of the power spectral density of the signal to be uniquely quantized.

This expression can be interpreted as the ratio between the arithmetic mean and the geometric mean of $S_X(f)$. In effect, if we consider the evaluation of $S_X(f)$ for N values in the interval $[-1/2, 1/2]$ or its equivalent in the interval $[0, 1]$, we find:

$$\int_0^1 S_X(f)\mathrm{d}f \approx \frac{1}{N}\sum_{k=0}^{N-1} S_X\left(\frac{k}{N}\right)$$

$$\exp\left(\int_0^1 \log_e S_X(f)\mathrm{d}f\right) \approx \exp\left(\frac{1}{N}\sum_{k=0}^{N-1} \log_e S_X\left(\frac{k}{N}\right)\right)$$

$$\exp\left(\int_0^1 \log_e S_X(f)\mathrm{d}f\right) \approx \exp\left(\log_e\left(\prod_{k=0}^{N-1} S_X\left(\frac{k}{N}\right)\right)^{1/N}\right)$$

$$\exp\left(\int_0^1 \log_e S_X(f)\mathrm{d}f\right) \approx \left(\prod_{k=0}^{N-1} S_X\left(\frac{k}{N}\right)\right)^{1/N}$$

The least predictable signal is white noise. The asymptotic value of prediction gain is equal to 1 as shown in equation [1.13]. The arithmetic and geometric means are equal. There is no hope of any gain while using predictive scalar quantization rather than standard scalar quantization.

Conversely, the most predictable signal is of the form:

$$X(n) = \sum_{k=1}^{K} a_k \cos(2\pi f_k n + \Phi_k)$$

where Φ_k are random phases. The prediction gain is infinite. Because the quantization error power, or the reconstruction error, is limited:

$$\sigma_Q^2 = c\frac{\sigma_X^2}{G_p(\infty)}\, 2^{-2b}$$

we see that a harmonic process can be quantized without distortion for whichever b are chosen. Evidently, this is purely theoretical since it says that we need to only code the different phases with a finite number of bits and that afterward there is no need to transmit any information for as long as they wish! The inverse ratio of the asymptotic value of the prediction gain is called spectral spread flatness.

1.3.5. *Closed-loop predictive scalar quantization*

Let us look at the diagram of the principle of predictive quantization in Figure 1.3. In this configuration, the quantizer requires the transmission at each instant n of the number $i(n)$, the result of the calculation of the prediction error $y(n)$, as well as of another number which is associated with the prediction quantization $v(n)$ itself. This quantization configuration, known as open-loop quantization configuration, is not realistic because we are not interested in multiplying the information to be encoded, at a constant resolution. The application of a *closed-loop* quantization is preferred as shown in Figure 1.4 since we can devote all the binary resources available to quantifying the prediction error $y(n)$. The transmission of $v(n)$ to the receiver is no longer necessary since $v(n)$ now represents the prediction of the reconstructed signal $\hat{x}(n)$. This prediction can be produced in an identical manner at the transmitter. All that is needed at the transmitter is a copy of the signal processing carried out at the receiver. We can speak of local decoding (at the transmitter) and of distance decoding

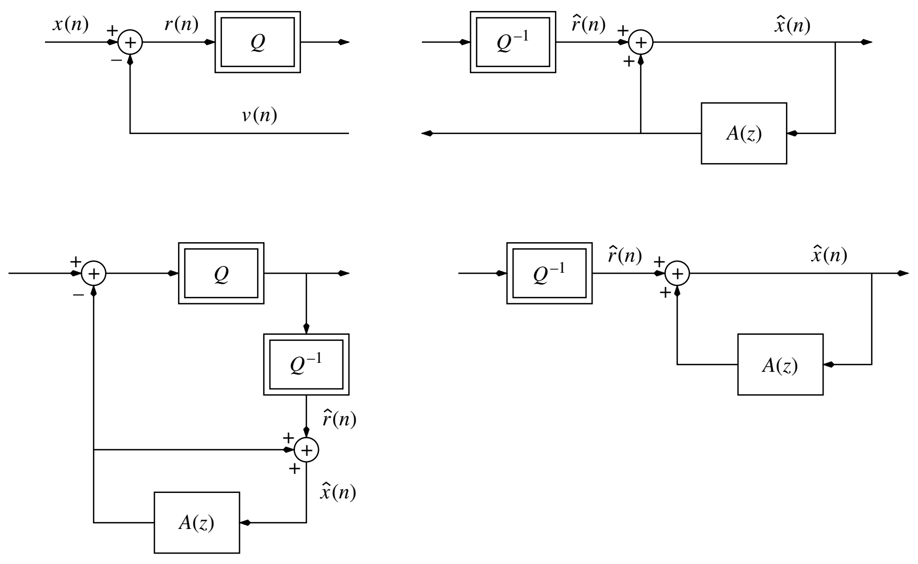

Figure 1.4. *Closed-loop predictive quantizer*

(at the receiver). This method of proceeding has a cost: the prediction is made on the reconstructed signal $\hat{x}(n)$ rather than on the original signal $x(n)$. This is not serious as long as $\hat{x}(n)$ is a good approximation of $x(n)$, when the intended compression rate is slightly increased.

We can now turn to the problem of determining the coefficients for the polynomial $A(z)$. The signals to be quantized are not time-invariant and the coefficients must be determined at regular intervals. If the signals can be considered to be locally stationary over N samples, it is enough to determine the coefficients for every N sample. The calculation is generally made from the signal $x(n)$. We say that the prediction is calculated *forward*. Although the information must be transmitted to the decoder, this transmission is possible as it requires a slightly higher rate. We can also calculate the filter coefficients from the signal $\hat{x}(n)$. In this case, we say that the prediction is calculated *backward*. This information does not need to be transmitted. The adaptation can even be made at the arrival of each new sample $\hat{x}(n)$ by a gradient algorithm (adaptive method).

Let us compare the details of the advantages and disadvantages of these two methods separately. *Forward* prediction uses more reliable data (this is particularly important when the statistical properties of a signal evolve rapidly), but it requires the transmission of *side information* and we must wait until the last sample of the current frame before starting the encoding procedure on the contents of the frame. We therefore have a reconstruction delay of at least N samples. With a *backward* prediction, the reconstruction delay can be very short but the prediction is not as good because it is produced from degraded samples. We can also note that, in this case, the encoding is more sensitive to transmission errors. This choice comes down to a function of the application.

We have yet to examine the problem of decoder filter stability because it is autoregressive. We cannot, in any case, accept the risk of instability. We have seen that if the autocovariance matrix estimate is made so as to maintain its definite-positive character, the filter stability is assured, and the poles of the transfer function are inside the unit circle.

Chapter 2

Vector Quantization

2.1. Introduction

When the resolution is low, it is natural to group several samples $x(n)$ in a vector $\underline{x}(m)$ and to find a way to quantize them together. This is known as vector quantization. The resolution b, the vector dimension N, and the size L of the codebook are related by:

$$L = 2^{bN}$$

In this case, b does not have to be an integer. The product bN must be an integer or even, more simply, L must be an integer. Vector quantization therefore enables the definition of non-integer resolutions. However, this is not the key property of vector quantization: vector quantization allows us to directly take account of the correlation contained in the signal rather than first decorrelating the signal, and then quantizing the decorrelated signal as performed in predictive scalar quantization. Vector quantization would be perfect were it is not for a major flaw: the complexity of processing in terms of the number of multiplications/additions to handle is an exponential function of N.

2.2. Rationale

Vector quantization is an immediate generalization of scalar quantization. Vector quantization of N dimensions with size L can be seen as an application of R^N in a finite set C which contains L N-dimensional vectors:

$$Q: \; R^N \longrightarrow C \quad \text{with} \quad C = \left\{ \underline{\hat{x}}^1 \cdots \underline{\hat{x}}^L \right\} \quad \text{where} \quad \underline{\hat{x}}^i \in R^N$$

The space R^N is partitioned into L regions or cells defined by:

$$\Theta^i = \{\underline{x} : Q(\underline{x}) = \underline{\hat{x}}^i\}$$

The codebook C can be compared with a matrix where necessary, and $\underline{\hat{x}}^i$ is the reproduction vector. We can also say that C represents the reproduction alphabet and $\underline{\hat{x}}^i$ represents the reproduction symbols in accordance with the usual vocabulary for information theory.

Equation [2.1] is a direct generalization of equation [1.1] which uses the probability density function and the Euclidean norm:

$$\sigma_Q^2 = \frac{1}{N}\sum_{i=1}^{L}\int_{\underline{u}\in\Theta^i} ||\underline{u} - \underline{\hat{x}}^i||^2 p_X(\underline{u})\mathrm{d}\underline{u} \qquad [2.1]$$

We need to determine the codebook C, that is, the reproduction vectors $\{\underline{\hat{x}}^1 \cdots \underline{\hat{x}}^L\}$ while minimizing σ_Q^2. This minimization leads us to the identical necessary conditions for optimization that we found for the scalar case:

– Given a codebook $C = \left\{\underline{\hat{x}}^1 \cdots \underline{\hat{x}}^L\right\}$, the best partition is that which satisfies:

$$\Theta^i = \{\underline{x} : ||\underline{x} - \underline{\hat{x}}^i||^2 \leq ||\underline{x} - \underline{\hat{x}}^j||^2 \quad \forall j \in \{1 \cdots L\}\ \}$$

This is the nearest neighbor rule. This partition is called the Voronoï partition.

– Given a partition, the best reproduction vectors are found using the centroid condition. For a quadratic distortion:

$$\hat{x}^i = \mathbb{E}\{\underline{X}|\underline{X} \in \Theta^i\} = \frac{\int_{\Theta^i} \underline{u} p_X(\underline{u})\mathrm{d}\underline{u}}{\int_{\Theta^i} p_X(\underline{u})\mathrm{d}\underline{u}}$$

The left-hand chart in Figure 2.1 shows a realization $x(n)$ of a random process AR(2) which is Gaussian and centered as a function of n. The right-hand chart is obtained from this realization by creating two-dimensional vectors ($N = 2$) such that $\underline{x}(m) = [x(2m)\ x(2m+1)]^t$ and projecting each vector $\underline{x}(m)$ in the plane. We obtain a cloud of points which has the tendency to align in proportion along the first diagonal as the first normalized autocovariance coefficient approaches 1.

The graphs in Figure 2.2 show two ways of partitioning the plane. The Voronoï partition corresponds to vector quantization which has been obtained by applying the generalized Lloyd–Max algorithm (see the following section) to the vector case with $b = 2$ and $N = 2$, that is, with the number of reproduction vectors L equal to 16. The partition which corresponds to scalar quantization, interpreted in the plane, shows rectangular-shaped elements and reproduction values which are positioned identically on the two axes. We can show that the ratio of the two axes of the ellipse on the second

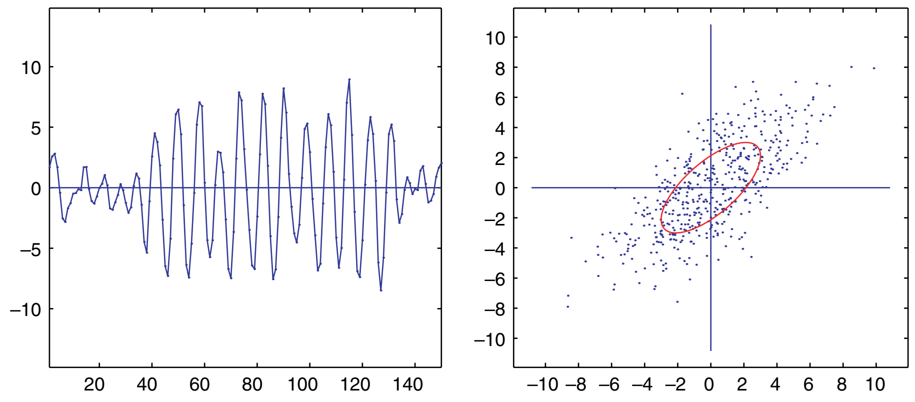
Figure 2.1. *Example of a realization of an AR(2) random process*

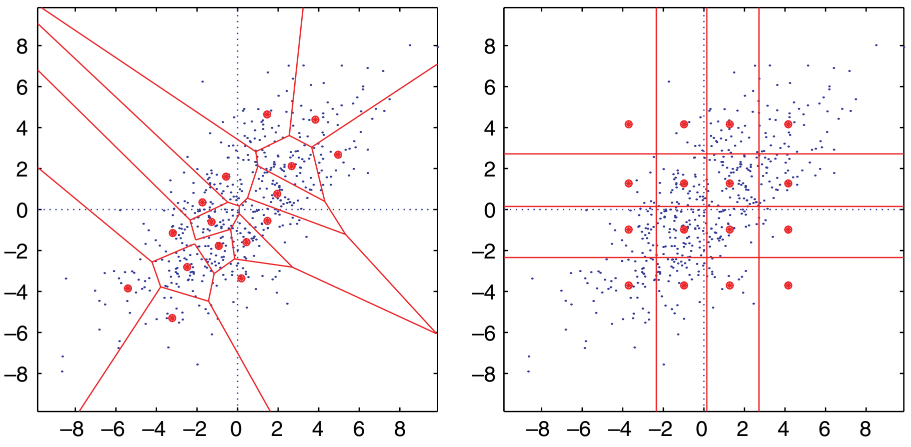
Figure 2.2. *Comparison of the performance of vector and scalar quantization with a resolution of 2. Vector quantization has L = 16 two-dimensional reproduction vectors. Scalar quantization has L = 4 reproduction values*

graph in Figure 2.1 is equal to $(1+\rho_1)/(1-\rho_1)$, where ρ_1 is the normalized covariance coefficient of order 1. From this we can deduce that the greater the correlation between the vector components, the more effective the vector quantization is since it adapts itself to the cloud configuration of the points while scalar quantization is scarcely modified. Vector quantization allows us to directly take account of the correlation contained in the signal rather than first decorrelating the signal and then quantizing the decorrelated signal as performed in predictive scalar quantization.

Figure 2.3 represents a sinusoidal process which is marred by noise. We can see clearly that vector quantization adapts itself much better to the signal characteristics.

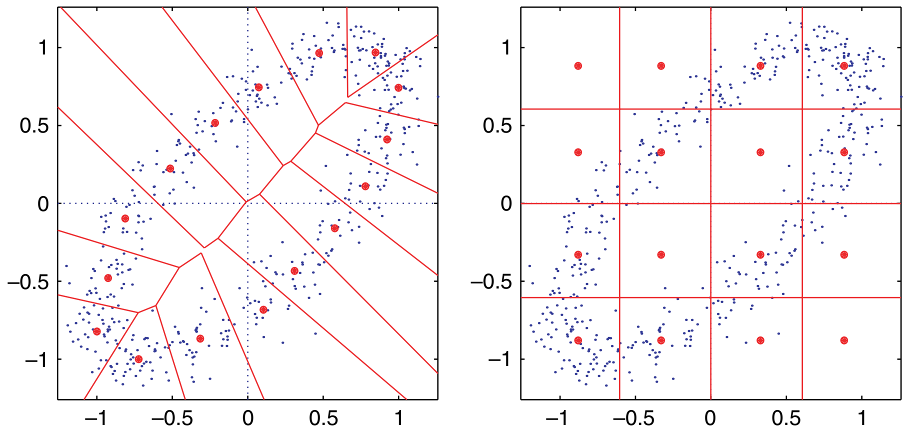

Figure 2.3. *Comparison of the performance of vector and scalar quantization for a sinusoidal process marred by noise*

A theorem, thanks to Shannon [GRA 90], shows that even for uncorrelated signals, from sources without memory (to use the usual term from information theory), gain is produced through vector quantization. This problem is strongly analogous to that of stacking spheres [SLO 84].

2.3. Optimum codebook generation

In practice, the probability density function $p_X(\underline{x})$ is unknown. We use empirical data (training data) for constructing a quantizer by giving each value the same weight. These training data must contain a large number of samples which are representative of the source. To create training data which are characteristic of speech signals, for example, we use several phonetically balanced phrases spoken by several speakers: male, female, young, old, etc.

We give here a summary of the Lloyd–Max algorithm which sets out a method for generating a quantizer. It is an iterative algorithm which successively satisfies the two optimization conditions.

– Initialize the codebook $\{\underline{\hat{x}}^1 \cdots \underline{\hat{x}}^L\}$, for example, by randomly generating it.

– From the codebook, $\{\underline{\hat{x}}^1 \cdots \underline{\hat{x}}^L\}$, label each sample in the training data with the number of its nearest neighbor, thus determining the optimum partition $\{\Theta^1 \cdots \Theta^L\}$ implicitly (explicit calculation is not necessary).

– For all the samples labeled with the same number, a new reproduction vector is calculated as the average of the samples.

– The mean distortion associated with these training data is calculated, and the algorithm ends when the distortion no longer decreases significantly, that is, when the reduction in the mean distortion is less than a given threshold, otherwise the two previous steps are repeated.

The decrease in the mean distortion is ensured; however, it does not always tend toward the global minimum but only reaches a local minimum. In fact, no theorems exist which prove that the mean distortion reaches a local minimum. New algorithms based on simulated annealing, for example, allow improvements (in theory) in quantizer performance.

Initializing the codebook presents a problem. The Linde-Buzo-Gray (LBG) algorithm [LIN 80], as it is known, is generally adopted, which can resolve this problem. The steps are as follows:

– First, a single-vector codebook which minimizes the mean distortion is found. This is the center of gravity of the training data. We write it as $\underline{\hat{x}}^0(b = 0)$. If the number of vectors in the training data is L', the distortion is:

$$\sigma_Q^2(b = 0) = \frac{1}{L'}\frac{1}{N}\sum_{m=0}^{L'-1} ||\underline{x}(m)||^2 = \sigma_X^2$$

since the signal is supposedly centered.

– Next, we *split* this vector into two vectors written $\underline{\hat{x}}^0(b = 1)$ and $\underline{\hat{x}}^1(b = 1)$ with $\underline{\hat{x}}^0(b = 1) = \underline{\hat{x}}^0(b = 0)$ and $\underline{\hat{x}}^1(b = 1) = \underline{\hat{x}}^0(b = 0) + \underline{\epsilon}$. Choosing the vector $\underline{\epsilon}$ presents a problem. We choose "small" values.

– Knowing that $\underline{\hat{x}}^0(b = 1)$ and $\underline{\hat{x}}^1(b = 1)$, we classify all the vectors in the training data relative to these two vectors (labeling all the vectors 0 or 1), and then calculate the new centers of gravity $\underline{\hat{x}}^0(b = 1)$ and $\underline{\hat{x}}^1(b = 1)$ of the vectors labeled 0 and 1, respectively.

– The distortion is calculated:

$$\sigma_Q^2(b = 1) = \frac{1}{L'}\frac{1}{N}\sum_{m=0}^{L'-1} ||\underline{x}(m) - \underline{\hat{x}}||^2$$

and this process is iterated as long as the decrease in the distortion is significant. Notice that the decrease is constantly ensured, thanks to the particular choice of initial vectors. In this way, the nearest neighbor rule and the centroid condition are applied a certain number of times to obtain the two reproduction vectors which minimize the mean distortion.

– We split these two vectors afresh into two, and so on.

– The algorithm is stopped when the desired number of vectors is reached.

2.4. Optimum quantizer performance

In the framework of the high-resolution hypothesis, Zador [ZAD 82] showed that the Bennett equation (scalar case),

$$\sigma_Q^2 = \frac{1}{12}\left(\int_R [p_X(x)]^{1/3}\mathrm{d}x\right)^3 \; 2^{-2b}$$

which gives the quantization error power as a function of the marginal probability density function of the process and the resolution, can be generalized to the vector case. We find:

$$\sigma_Q^2(N) = \alpha(N)\left(\int_{R^N} [p_X(\underline{x})]^{N/(N+2)}\mathrm{d}\underline{x}\right)^{(N+2)/N} \; 2^{-2b}$$

where $p_X(\underline{x})$ now represents the probability density function and $\alpha(N)$ is a constant which depends only on N. In the Gaussian case:

$$\sigma_Q^2(N) = \alpha(N)\left((2\pi)^{N/(N+2)}\left(\frac{N+2}{N}\right)^{N/2}(\det\mathbb{R})^{1/(N+2)}\right)^{(N+2)/N} \; 2^{-2b}$$

$$\sigma_Q^2(N) = \alpha(N)2\pi\left(\frac{N+2}{N}\right)^{(N+2)/2}(\det\mathbb{R})^{1/N} \; 2^{-2b}$$

such that:

$$\sigma_Q^2(N) = c(N)(\det\mathbb{R})^{1/N} \; 2^{-2b}$$

We show that:

$$c(1) = \frac{\sqrt{3}}{2}\pi > c(N) > c(\infty) = 1$$

When $N = 1$, we find equation [1.5] again.

As in the case of predictive scalar quantization, we can assess the performance improvement brought about by vector quantization relative to scalar quantization. The *vector quantization gain* is defined similarly to [1.11] and comprises two terms:

$$G_v(N) = \frac{c(1)}{c(N)} \times \frac{\sigma_X^2}{(\det\mathbb{R})^{1/N}}$$

The ratio $c(1)/c(N)$ is always greater than 1 which shows that, even for a source without memory, vector quantization is preferred but this contribution is limited because:

$$10\log_{10}\frac{c(1)}{c(N)} < 10\log_{10}\frac{c(1)}{c(\infty)} = 4.35 \text{ d}B$$

The second ratio represents how vector quantization takes account of the correlation between the different vector components. When $N \to \infty$, the ratio tends toward the asymptotic prediction gain value $G_p(\infty)$ as shown in equation [1.12].

Figure 2.4 shows the signal-to-noise ratio for vector quantization (as a function of N) and for predictive scalar quantization (as a function of $P + 1$) for $b = 2$. The limit of the signal-to-noise ratio for vector quantization can be seen when N tends toward infinity. The signal-to-noise ratio for predictive scalar quantization is:

$$\text{SNR}_{\text{QSP}} = 6.02\, b - 4.35 + 10 \log_{10} G_p(\infty)$$

when $P \geq 2$. The 4.35-dB shift between the two horizontal lines is from the ratio $c(1)/c(\infty)$. Vector quantization offers a wide choice in the selection of the geometric shape of the partition. This explains the gain of 4.35 dB (when N tends toward infinity).

Vector quantization makes direct use of correlation in the signal. Predictive scalar quantization also uses this correlation but only after decorrelating the signal.

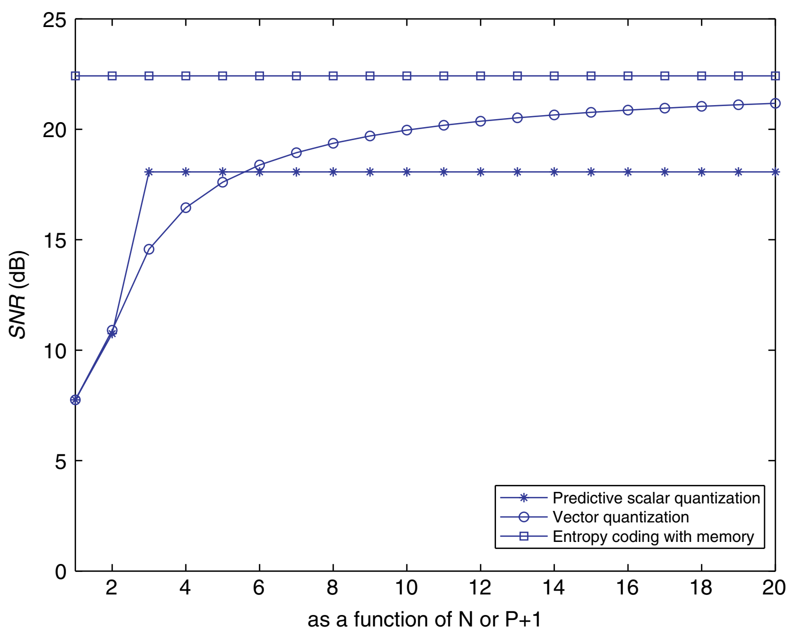

Figure 2.4. *Signal-to-noise ratio as a function of N for vector quantization and as a function of P + 1 for predictive scalar quantization*

As soon as N is greater than a relatively low value, vector quantization performs better than predictive scalar quantization. As N increases, the performance of vector quantization rapidly approaches the limit for a time-invariant process. It can be shown that no quantizer is capable of producing a signal-to-noise ratio better than this limit. Vector quantization is therefore considered to be the optimum quantization, provided that N is sufficiently large.

2.5. Using the quantizer

In principle, quantizing a signal involves regrouping the samples of the signal to be compressed into a set of N-dimensional vectors, applying the nearest neighbor rule to find the vector's number for encoding and extracting a vector from a table to an address given to supply the reproduction vector for decoding. In practice, a whole series of difficulties may arise, usually due to the processor's finite calculation power when processing; both encoding and decoding must generally be performed in real time. In fact, the most intensive calculations come from encoding since decoding the processor only needs to look up a vector at a given address in a table.

Let us take the example of telephone-band speech with a resolution $b = 1$. We want to perform an encoding at, for example, 8 kbit/s. We must answer the question of how to choose the vector dimension N and the number L of vectors which form the codebook. We have just seen that it is in our interests to increase N but the size of the codebook increases exponentially since $L = 2^{bN}$. The calculation load (the number of multiplications–accumulations) also increases exponentially with $NL = N2^{bn}$ with 2^{bN} per sample or $2^{bN} f_e$ multiplications–accumulations per second. We can assume that current signal processors can handle around 10^8 multiplications–accumulations per second. Therefore, we must have:

$$2^{bN} \times 8 \times 10^3 \leq 10^8$$

which leads to $N \leq 13$ for $b = 1$. This is too low for at least two reasons. On the one hand, speech signals are too complex to summarize in 2^{13} vectors. On the other hand, the autocorrelation function does not cancel itself out in a few dozen samples. Vector quantization does not require that the components of the vector to be quantized are decorrelated (intraframe correlation) since it adapts correctly to this correlation. However, the vectors themselves must be as decorrelated as possible (interframe decorrelation). When $N = 13$, the interframe correlation is still significant for speech signals.

N and L can be increased without changing the calculation load by setting up a particular codebook structure. Numerous propositions have been made and a highly detailed presentation of these can be found in [GER 92]. Here, we present a summary of the numerous possibilities.

2.5.1. *Tree-structured vector quantization*

Take a collection of codebooks, each containing two vectors (binary tree case). In the first stage, we select the nearest neighbor to $\underline{x}(m)$ in the first codebook. In the second stage, we select the nearest neighbor to $\underline{x}(m)$ in one or other of the two codebooks depending on the first choice, etc. The calculation cost per sample for a standard codebook is 2^{bN}. This reduces to 2^{bN} when a binary tree structure is used. The LBG algorithm is well adapted to constructing this codebook since it supplies all the intermediate codebooks (with a little adaptation). Note that the decoder uses only the codebook of size 2^{bN}.

2.5.2. *Cartesian product vector quantization*

A vector is decomposed into several sub-vectors which eventually have different dimensions and a vector quantization is applied individually to each sub-vector. We therefore lose the use of the statistical dependence between the vector components. A method of decomposing the vector so that no dependence exists between the sub-parts is required. The complexity becomes proportional to $\sum_i L_i$, where L_i is the number of vectors in the codebook C_i.

2.5.3. *Gain-shape vector quantization*

This is a special case of the previous case. The idea involves separating the shape characteristics, the spectral content of the vector, and the gain characteristics, the vector energy, into two distinct codebooks. The vector codebook is therefore composed of vectors which are (eventually) normalized. Let us take speech signals as an example. The spectrum content is slightly modified when speech is soft or loud. However, the energy in the signal is strongly amplified. These two types of information are highly decorrelated and therefore justify the use of this method.

2.5.4. *Multistage vector quantization*

Successive approximations are used in this process. First, a coarse vector quantization is carried out, then a second quantization on the quantization error, etc. as illustrated by Figure 2.5.

2.5.5. *Vector quantization by transform*

An initial orthogonal transform is applied followed by a vector quantization on the transformed vector. Since an orthogonal transformation does not change the vector norm, the meansquared error is unchanged. *A priori*, this transformation is

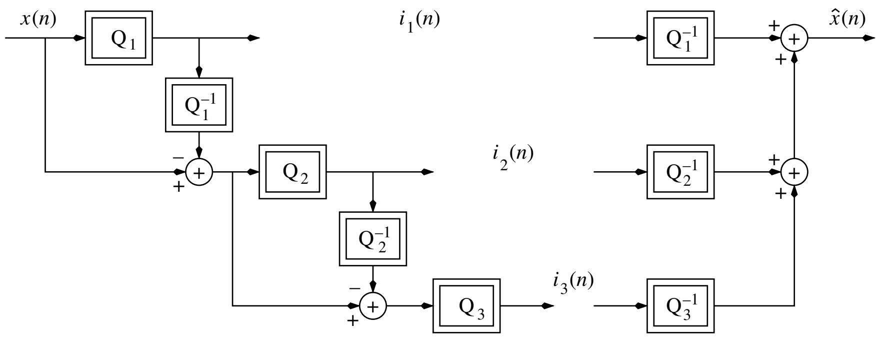

Figure 2.5. *Multistage scalar or vector quantization*

not very interesting. To encode signals, a initial transformation is often applied followed by quantization in the transformed domain, but this quantization is usually scalar. An entire chapter is dedicated to this subject. It is sometimes useful to associate a transformation and vector quantization to concentrate information in a lower dimensional vector by neglecting low-energy components. The reduction in complexity that comes about from this truncation can be important. Some classic orthogonal transformations in signal processing include, for example, the discrete Fourier transform and the discrete cosine transform, which has the advantage, in relation to the previous method, of being real-valued.

2.5.6. *Algebraic vector quantization*

In this method, the codebook is no longer constructed using the Lloyd–Max algorithm. It is independent of the source statistics. The reproduction vectors are regularly distributed in space. This is known as trellis vector quantization. These codebooks are widely used since they reduce the calculation load significantly and they need not be memorized.

2.6. Gain-shape vector quantization

One particular type of vector quantization, known as gain-shape quantization, is widely used, especially for audio signal coding, since this vector quantization can manage the time evolution of the instantaneous power of a sound source conveniently. Rather than minimizing the norm $||\underline{x} - \underline{\hat{x}}^j||$ relative to j, $||\underline{x} - \hat{g}^i \underline{\hat{x}}^j||$ is minimized relative to i and j. We show in this section how this is a modification of the nearest neighbor rule and how to construct the optimum codebook.

2.6.1. *Nearest neighbor rule*

Two codebooks are used as shown in Figure 2.6. The first codebook $\left\{\underline{\hat{x}}^1 \cdots \underline{\hat{x}}^{L_1}\right\}$ comprises L_1 N-dimensional vectors which are possibly normed, and the second codebook $\left\{\hat{g}^1 \cdots \hat{g}^{L_2}\right\}$ comprises L_2 scalars. Minimizing the squared error relative to the two indices i and j can be done through an exhaustive search. However, the preferred method to find the minimization is, at the expense of making an approximation, done in two steps. First, the vector in the first codebook which most closely represents $\underline{x}$ is found. Next, the gain is calculated using a scalar method.

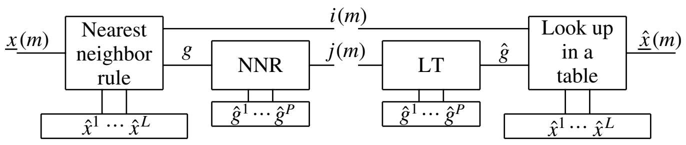

Figure 2.6. *Gain-shape vector quantizer*

Let us consider the system depicted in Figure 2.7.

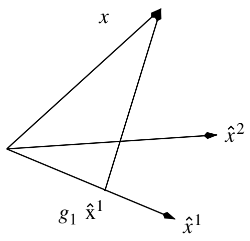

Figure 2.7. *Gain-shape vector quantization*

The gain, $g(j)$, is defined so that the vector $g(j)\,\underline{\hat{x}}^j$ is the orthogonal projection of $\underline{x}$ on $\underline{\hat{x}}^j$. This gain is given by:

$$<\underline{x} - g(j)\underline{\hat{x}}^j, \underline{\hat{x}}^j >= 0$$

$$g(j) = \frac{<\underline{x}, \underline{\hat{x}}^j>}{||\underline{\hat{x}}^j||^2}$$

where $<\underline{x}, \underline{y}>$ is the scalar product of the two vectors $\underline{x}$ and $\underline{y}$. Since:

$$\min_j ||\underline{x} - g(j)\underline{\hat{x}}^j||^2 \equiv \min_j (||\underline{x}||^2 - 2g(j) <\underline{x}, \underline{\hat{x}}^j> + g^2(j)||\underline{\hat{x}}^j||^2)$$

$$\min_j ||\underline{x} - g(j)\underline{\hat{x}}^j||^2 \equiv \min_j (||\underline{x}||^2 - \frac{<\underline{x}, \underline{\hat{x}}^j>^2}{||\underline{\hat{x}}^j||^2})$$

by substituting in the value for $g(j)$, the minimization relative to j becomes a maximization:

$$\min_j ||\underline{x} - g(j)\underline{\hat{x}}^j||^2 \equiv \max_j \frac{<\underline{x}, \underline{\hat{x}}^j>^2}{||\underline{\hat{x}}^j||^2} \equiv \max_j <\underline{x}, \frac{\underline{\hat{x}}^j}{||\underline{\hat{x}}^j||}>^2 \equiv \max_j |cos\phi^j|$$

where ϕ^j represents the angle between $\underline{x}$ and $\underline{\hat{x}}^j$. The nearest neighbor rule involves therefore, in this case, choosing the vector which is at the lowest angle (to within π) to $\underline{x}$ from all the possible vectors.

2.6.2. *Lloyd–Max algorithm*

We now show how to construct the first codebook, which is characteristic of the shape. We have seen that the Lloyd–Max algorithm is made of two stages. Knowing the training data of M vectors $\underline{x}(m)$ and an initial codebook made of L_1 vectors $\underline{\hat{x}}^j$, which are assumed to be normalized without losing any generality, we need to find the best partition of M vectors into L_1 classes. Each vector in the training data is labeled by the index j which maximizes $<\underline{x}(m), \underline{\hat{x}}^j>^2$. Knowing this partition, the rest of the codebook can then be determined. From the $M(j)$ vectors $\underline{x}(m)$, which are associated with the previous vector $\underline{\hat{x}}^j$, the new vector $\underline{\hat{x}}^j$ which minimizes the meansquared error is calculated:

$$\sigma_Q^2(j) = \frac{1}{M(j)} \left[\sum_{m=1}^{M(j)} ||\underline{x}(m)||^2 - \sum_{m=1}^{M(j)} <\underline{x}(m), \underline{\hat{x}}^j>^2 \right]$$

Similarly, the expression:

$$Q = \sum_{m=1}^{M(j)} <\underline{x}(m), \underline{\hat{x}}^j>^2 = (\underline{\hat{x}}^j)^t \Gamma \underline{\hat{x}}^j$$

is maximized with regard to $\underline{\hat{x}}^j$ where Γ is the empirical covariance matrix:

$$\Gamma = \sum_{m=1}^{M(j)} \underline{x}(m)\underline{x}^t(m)$$

Maximizing Q within the constraint $||\underline{\hat{x}}^j|| = 1$ is achieved through the Lagrange multipliers method. The derivative of:

$$(\underline{\hat{x}}^j)^t \Gamma \underline{\hat{x}}^j - \lambda[(\underline{\hat{x}}^j)^t \underline{\hat{x}}^j - 1]$$

with respect to $\underline{\hat{x}}^j$ is canceled out. We find:

$$\Gamma \underline{\hat{x}}^j = \lambda \underline{\hat{x}}^j$$

The parameter λ is therefore an eigenvalue of the matrix Γ. By multiplying this equation by $(\underline{\hat{x}}^j)^t$, we find:

$$(\underline{\hat{x}}^j)^t \Gamma \underline{\hat{x}}^j = \lambda (\underline{\hat{x}}^j)^t \underline{\hat{x}}^j = \lambda$$

Since the first term is the expression that we seek to maximize, we must choose the eigenvector which is associated with the largest eigenvalue in the empirical covariance matrix. This is a classic result of principal component analysis (PCA) [JAY 84].

Chapter 3

Sub-band Transform Coding

3.1. Introduction

Until very recently, a classic distinction was made between transform coding and sub-band coding. In a transform coder, an N-dimensional vector is first determined from the signal. Next, a generally orthogonal or unitary matrix is applied (for example, a matrix constructed from the discrete Fourier transform or the discrete cosine transform), then the coefficients are encoded in the transformed domain. At the receiver, the reverse transform is applied. In a sub-band encoder, the signal is first filtered by a bank of analysis filters, then each sub-band signal is downsampled and the downsamples are encoded separately. At the receiver, each component is decoded, an upsampling and interpolation by a bank of synthesis filters is carried out, then the separate reconstituted signals are added together. These two encoding techniques are equivalent, as will be shown in section 3.2. For further details, refer to the theory of perfect reconstruction filter banks [MAL 92, VAI 90], multi-resolution systems, and wavelet transforms [RIO 93].

An important issue in a coding system is the allocation of available bits to different sub-sets. Two extremes can be imagined: distributing bits over several heterogeneous parts, for example, over the filter coefficients and the excitation signal in speech coding, or distributing bits between different elements which present *a priori* the same distribution, for example, successive samples in a signal. In Part 2 of this book, we will see that transform (or sub-band) coding techniques involve making the different vector components non-homogeneous in such a way that more bits can be reserved for the more significant components. We will show that the optimum transformation, the Karhunen–Loeve transform, decorrelates the components of the transformed vector. This transform concentrates the highest signal power into the lowest possible number of components.

We will also see that minimizing the mean squared error (MSE), the criteria with which we are concerned at present, has the effect of making the quantization noise as white as possible. This approach is somewhat theoretical: in practice, the issue is not really aiming at making the quantization noise as low or as white as possible. Rather, we look to constrain the shape of the noise spectrum so that it fits within psychoacoustic or psychovisual constraints. This is known as "perceptual" coding. This question is not addressed in this chapter; it will be broached in Part 2 of the book.

3.2. Equivalence of filter banks and transforms

Consider a bank of M filters with scalar quantization of sub-band signals as shown in Figure 3.1. Assume that the filters are all finite impulse response filters, causal, with the same number of coefficients N. The downsampling factor is M' and $b_0 \cdots b_{M-1}$ are the numbers of bits assigned to the quantization of each sub-band signal. When $M' = M$, quantization in the transform domain has the same number of samples as it would have if quantization had been carried out in the time domain. At this point, downsampling is *critical*, the filter bank is at *maximum decimation*. This is the usual choice in encoding (our assumption is verified as follows).

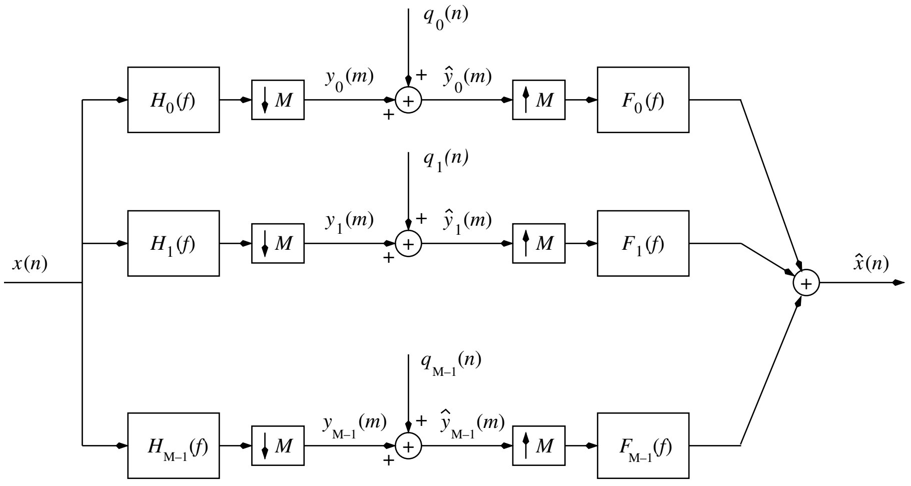

Figure 3.1. *Diagram of a sub-band coding system (with critical downsampling)*

By constructing the vector $\underline{x}(m) = [x_0(m) \cdots x_{N-1}(m)]^t$ from its polyphase components $x_l(m) = x(mM - l)$ and the vector $\underline{y}(m) = [y_0(m) \cdots y_{M-1}(m)]^t$

from the sub-band signals, the M equations:

$$y_k(m) = \sum_{l=0}^{N-1} h_k(l)x(mM - l)$$

can be written in matrix form:

$$\underline{y}(m) = \begin{bmatrix} h_0(0) & \cdots & h_0(N-1) \\ \vdots & \ddots & \vdots \\ h_{M-1}(0) & \cdots & h_{M-1}(N-1) \end{bmatrix} \underline{x}(m) = \mathcal{T}\,\underline{x}(m) \qquad [3.1]$$

The analysis filter bank is therefore equivalent to a transformation $\mathcal{T}$ which is characterized by a rectangular matrix $M \times N$ in which the line vectors are the analysis filter impulse responses. We can also show that a bank of synthesis filters is equivalent to a transformation $\mathcal{P}$ which is characterized by a rectangular matrix $N \times M$ in which the reversed column vectors are analysis filter impulse responses. The general condition for perfect reconstruction, the *bi-orthogonality* condition, is simply that the matrix multiplication $\mathcal{PT}$ results in the identity matrix when $N = M$ (in the trivial case). In the usual case where $N > M$ (non-trivial case), the bi-orthogonality condition is generalized [VAI 90].

Finding impulse responses $h_k(n)$ and $f_k(n)$ which satisfy this condition is a difficult problem. To simplify the problem, we usually[1] do with *modulated* filter banks by introducing a strong constraint on $h_k(n)$ and $f_k(n)$. These impulse responses are obtained from prototype filter impulse responses $h(n)$ and $f(n)$ via modulation: $h_k(n) = h(n)d_k(n)$ and $f_k(n) = f(n)d_k(n)$ for $k = 0 \cdots M-1$ and $n = 0 \cdots N-1$. Note that equation [3.1] becomes:

$$\underline{y}(m) = D\,\underline{u}(m) \quad \text{with} \quad \underline{u}(m) = \underline{h} \otimes \underline{x}(m)$$

where D is the modulation matrix and the operation $\otimes$ involves multiplying the two vectors component by component. The vector $\underline{u}(m)$ is simply the version of the vector $\underline{x}(m)$ weighted by the weighting window $\underline{h}$.

$d_k(n)$, $h(n)$, and $f(n)$ must be determined. Starting from an ideal low-pass prototype filter with frequency cut-off $1/4M$, and constructing from this prototype filter the frequency response $H_0(f) \cdots H_{M-1}(f)$ of the M bank filters as follows:

$$H_k(f) = H\left(f - \frac{2k+1}{4M}\right) + H\left(f + \frac{2k+1}{4M}\right)$$

we can directly deduce the relation between the impulse responses:

$$h_k(n) = 2\,h(n)\cos\left(2\pi\frac{2k+1}{4M}n\right)$$

1. Except in the case where wavelet transforms are used, especially in image coding.

We can find equivalent relations for the synthesis filter bank. In practice, it is impossible to prevent overlapping in the frequency domain as a result of the finite nature of N. Similarly, solutions to the problem of perfect reconstruction do exist. The classic solution is the modified discrete cosine transform (MDCT, TDAC, MLT, etc.), where the matrix D is a cosine matrix with suitable phases:

$$d_k(n) = \sqrt{\frac{2}{M}} \cos\left[(2k+1)(2n+1+N-M)\frac{\pi}{4M}\right]$$

for $k = 0 \cdots M-1$ and $n = 0 \cdots N-1$.

We can show that the perfect reconstruction condition is written as:

$$h^2(n) + h^2(n+M) = 1 \quad \text{for } n = 0 \cdots M-1 \qquad [3.2]$$

with:

$$f(n) = h(n) \quad \text{for } n = 0 \cdots N-1$$

and this condition is satisfied by:

$$h(n) = f(n) = \sin\left[(2n+1)\frac{\pi}{4M}\right]$$

This transform is well used (in audio coding).

All that remains for us to do is to determine N and M. This is a delicate question in practice since the transform must satisfy opposing constraints. We generally want good frequency properties (low overlapping in the frequency domain, good frequency resolution) without negatively affecting the time resolution (to handle rapid variations in the instantaneous signal power). This question will be addressed in the second part of the book.

3.3. Bit allocation

3.3.1. *Defining the problem*

The M components $y_k(m)$ of the vector $\underline{y}(m) = \mathcal{T}\underline{x}(m)$ can be interpreted as the realization of M time-invariant random processes $Y_k(m)$ with power $\sigma^2_{Y_k}$. The quantization error powers are written as $\sigma^2_{Q_0} \cdots \sigma^2_{Q_{M-1}}$. By admitting that the perfect reconstruction condition involves conserving the power as in the case of an orthogonal or unitary transform, we deduce that the reconstruction error power (which is of interest to the user):

$$\sigma^2_{\bar{Q}} = \mathbb{E}\{|X(n) - \hat{X}(n)|^2\} = \frac{1}{N}\mathbb{E}\{||\underline{X}(m) - \underline{\hat{X}}(m)||^2\}$$

is equal to the arithmetic mean of the quantization error powers of the different components (errors introduced locally by the quantizers):

$$\sigma_Q^2 = \frac{1}{M}\mathbb{E}\{||\underline{Y}(m) - \hat{\underline{Y}}(m)||^2\} = \frac{1}{M}\sum_{k=0}^{M-1}\sigma_{Q_k}^2$$

The first problem that we examine is as follows. We apply an ordinary transform $\mathcal{T}$. We have M bits at our disposal to scalar quantize the M coefficients, the M sub-band signals. We need to find the optimum way of distributing the bM binary resources available. This requires, therefore, that we determine the vector $\underline{b} = [b_0 \cdots b_{M-1}]^t$ while minimizing σ_Q^2 within the constraint:

$$\sum_{k=0}^{M-1} b_k \leq bM$$

3.3.2. *Optimum bit allocation*

In the framework of the high-resolution hypothesis, we can express $\sigma_{Q_k}^2$ as a function of $\sigma_{Y_k}^2$ and b_k by applying [1.5]. Recall that the expression underlines the fact that $\sigma_{Q_k}^2$ depends on b_k:

$$\sigma_{Q_k}^2(b_k) = c_k(1)\,\sigma_{Y_k}^2\,2^{-2b_k} \qquad [3.3]$$

where $c_k(1)$ is a constant which depends only on the marginal probability density function of $Y_k(m)$. The repercussions of the hypothesis of Gaussian form on the sub-band signals are that all the constants $c_k(1)$ are equal. This common constant is known as $c(1)$.

Let us show that the optimum bit allocation which minimizes:

$$\sigma_Q^2 = \frac{c(1)}{M}\sum_{k=0}^{M-1}\sigma_{Y_k}^2\,2^{-2b_k} \qquad [3.4]$$

within the constraint $\sum_{k=0}^{M-1} b_k \leq bM$ is given by:

$$b_k = b + \frac{1}{2}\log_2\frac{\sigma_{Y_k}^2}{\alpha^2} \qquad [3.5]$$

where b is the mean number of bits per component and:

$$\alpha^2 = \left(\prod_{k=0}^{M-1}\sigma_{Y_k}^2\right)^{1/M}$$

is the geometric mean of the powers.

In effect, we know that as long as we have M positive real numbers, the arithmetic mean is greater than or equal to the geometric mean and that this equivalence occurs when the M numbers are equal as follows:

$$\frac{1}{M}\sum_{k=0}^{M-1} a_k \geq \left(\prod_{k=0}^{M-1} a_k\right)^{1/M}$$

If we put $a_k = \sigma_{Y_k}^2 2^{-2b_k}$, this relation becomes:

$$\frac{1}{M}\sum_{k=0}^{M-1} \sigma_{Y_k}^2\, 2^{-2b_k} \geq \left(\prod_{k=0}^{M-1} \sigma_{Y_k}^2 2^{-2b_k}\right)^{1/M} = \left(\prod_{k=0}^{M-1} \sigma_{Y_k}^2\right)^{1/M} 2^{-2\sum_{k=0}^{M-1} b_k/M}$$

$$\frac{1}{M}\sum_{k=0}^{M-1} \sigma_{Y_k}^2 2^{-2b_k} \geq \alpha^2\, 2^{-2b}$$

The optimum value is achieved when all the intermediate terms in the sum are equal. In this case, the inequality becomes an equality. Whatever the value of k is, we have:

$$\sigma_{Y_k}^2\, 2^{-2b_k} = \alpha^2\, 2^{-2b} \qquad [3.6]$$

such that:

$$2^{b_k} = 2^b \sqrt{\frac{\sigma_{Y_k}^2}{\alpha^2}} \qquad [3.7]$$

Hence, we find equation [3.5].

Equation [3.6] has an interesting interpretation. It means that the optimum scalar quantization of each random variable $Y_k(m)$ gives rise to the same MSE. We therefore arrive at the following important conclusion: everything happened as though we were seeking to make the reconstruction error as close as possible to a full-band white noise.

The quantization error power can be written as:

$$\sigma_Q^2 = c(1)\left(\prod_{k=0}^{M-1} \sigma_{Y_k}^2\right)^{1/M} 2^{-2b} \qquad [3.8]$$

3.3.3. *Practical algorithm*

Equation [3.5] is not directly usable. In effect, the number of bits b_k obtained is not always an integer number, nor is it guaranteed to be a positive number. One solution involves using a sub-optimal algorithm [2] which progressively distributes the bits to where they have the greatest effect [GER 92]:

```
– Initialization
    - $b_0 = \cdots = b_{M-1} = 0$
    - $\sigma^2_{Q_0} = \sigma^2_{Y_0} \cdots \sigma^2_{Q_{M-1}} = \sigma^2_{Y_{M-1}}$
– As long as $\sum_{k=0}^{M-1} b_k < bM$
    - $l = \arg\max_k \sigma^2_{Q_k}$
    - $b_l = b_l + 1$
    - $\sigma^2_{Q_l} = \sigma^2_{Q_l}/4$
```

3.3.4. *Further information*

In the standard approach to optimal bit allocation as shown earlier, $\sigma^2_{Q_k}$ is written as a function of $\sigma^2_{Y_k}$ and of the number of bits b_k which are dedicated to quantizing the sub-band signal $Y_k(m)$. Equation [3.3] restates that this is within the high-resolution hypothesis. This hypothesis is too constraining in real-world applications.

Assume that for each sub-band signal, we have a certain number of possible quantizers available and that the selected quantizer is written $Q_k(i_k)$. We choose, for example, to quantize each signal $Y_k(m)$ at different resolutions by a uniform scalar quantizer. In this very simple case, the number i_k is the resolution b_k. Assume also that for each quantizer we can deduce (or measure) the quantization error power $\sigma^2_{Q_k}(i_k)$ that it generates. The number of bits that it requires is $b_k(i_k)$. We will see in Chapter 4 that an entropy coding can be carried out after a uniform quantization. In this case, we show that the necessary number of bits required to quantize the signal can be reduced, which explains the notation $b_k(i_k)$ and the fact that $b_k(i_k)$ can be a non-integer.

If we create any combination of M quantizers:

$$\underline{i} = [i_0, \ldots, i_{M-1}]^t$$

from all the possible combinations, the quantization error power is calculated:

$$\sigma^2_Q(\underline{i}) = \frac{1}{M}\sum_{k=0}^{M-1} \sigma^2_{Q_k}(i_k) \qquad [3.9]$$

2. The algorithm is described as *greedy*, a term which is used every time that the optimum choice is made at each algorithm step in the hope of obtaining a globally optimized result.

and the average number of bits necessary to quantize the M signals is:

$$b(\underline{i}) = \frac{1}{M} \sum_{k=0}^{M-1} b_k(i_k) \qquad [3.10]$$

The two results are represented by a point in the plane σ_Q^2 as a function of b. The set of these points forms a cloud which is not unlike that shown in Figure 3.2.

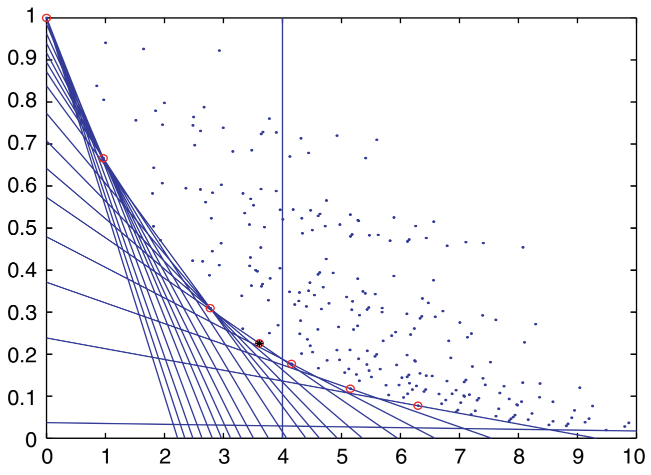

Figure 3.2. *Set of points with coordinates* $(\sigma_Q^2(\underline{i}), b(\underline{i}))$

The bit rate–distortion theory presented (very succinctly) in Chapter 4 will show more than likely that a convex envelope exists. We look for the point in the cloud, or of the convex envelope, which minimizes $\sigma_Q^2(\underline{i})$ within the constraint that $b(\underline{i})$ is less than or equal to a desired value b_d. This procedure is carried out twice [SHO 88]. First, we look for a point on the convex envelope as a function of a parameter λ, then we determine the optimum value of this parameter.

Let us examine the first optimization that is on the right-hand side of the equation $\sigma_Q^2 = -\lambda b + \alpha$. For a given slope $-\lambda$, we look for the line, that is, the parameter α, which passes through a point in the cloud and intersects the axis σ_Q^2 as low as possible. Because $\alpha(\underline{i}) = \sigma_Q^2(\underline{i}) + \lambda b(\underline{i})$, the point that we are looking for in the cloud is characterized by:

$$\underline{i}_{\text{opt}}(\lambda) = \arg\min_{\underline{i}} [\sigma_Q^2(\underline{i}) + \lambda b(\underline{i})]$$

Using equations [3.9] and [3.10], we find:

$$\underline{i}_{\text{opt}}(\lambda) = \arg\min_{\underline{i}} \left[\frac{1}{M} \sum_{k=0}^{M-1} \sigma^2_{Q_k}(i_k) + \lambda \frac{1}{M} \sum_{k=0}^{M-1} b_k(i_k) \right]$$

such that:

$$\underline{i}_{\text{opt}}(\lambda) = \arg \sum_{k=0}^{M-1} \min_{i_k} [\sigma^2_{Q_k}(i_k) + \lambda b_k(i_k)]$$

We note that realizing M independent minimizations is enough to avoid the necessity of examining all possible combinations of different quantizers. Take the case where we would need to select one quantizer out of 15, each of which has 32 sub-bands as we will see later in the case of audio coding via MPEG-1. The previous comment is not insignificant because we go from 15^{32} comparisons to 15×32.

We now proceed to the second optimization. All the lines determined by the minimization before are characterized by equations which no longer depend on λ:

$$\alpha[\underline{i}_{\text{opt}}(\lambda)] = \sigma^2_Q + \lambda b \qquad [3.11]$$

From all possible lines, we need only to choose the one that maximizes the y-coordinate of the intersection point with the line $b = b_d$. This coordinate is written as:

$$\beta(\lambda) = \alpha[\underline{i}_{\text{opt}}(\lambda)] - \lambda b_d$$

that is:

$$\beta(\lambda) = \sigma^2_Q[\underline{i}_{\text{opt}}(\lambda)] + \lambda b[\underline{i}_{\text{opt}}(\lambda)] - \lambda b_d$$

we need to determine only the value of λ that maximizes:

$$\lambda_{\text{opt}} = \arg\max_{\lambda} [\sigma^2_Q[\underline{i}_{\text{opt}}(\lambda)] + \lambda b[\underline{i}_{\text{opt}}(\lambda)] - \lambda b_d]$$

The M components of $i_{\text{opt}}(\lambda_{\text{opt}})$ characterize the M optimal quantizers.

Consider the following example. Assume that an orthogonal transform decomposes a signal into $M = 4$ sub-bands. The sub-band signal powers $\sigma^2_{Y_k}$ are simply obtained via random generation. Imagine that we have four possible sub-band quantizers available. The quantizers are uniform quantizers and are followed by entropy coding. The powers $\sigma^2_{Q_k}(i_k)$ are determined by calculating as if equation [3.3] was valid with $b_k \in \{0, \ldots, 3\}$. To simulate entropy coding and to obtain the expressions for $b_k(i_k)$, the value of b_k is randomly reduced a little. Figure 3.2 illustrates the collection of points whose coordinates were calculated from equations [3.9] and [3.10] in the same way as equation [3.11]. Four quantizers must be found which generate a lower abscissus that $b_d = 4$ and make σ^2_Q as small as possible. Figure 3.3 shows that the function $\beta(\lambda)$ is concave. A simple gradient calculation can be used to obtain λ_{opt}. From this, we deduce the M optimum quantizers.

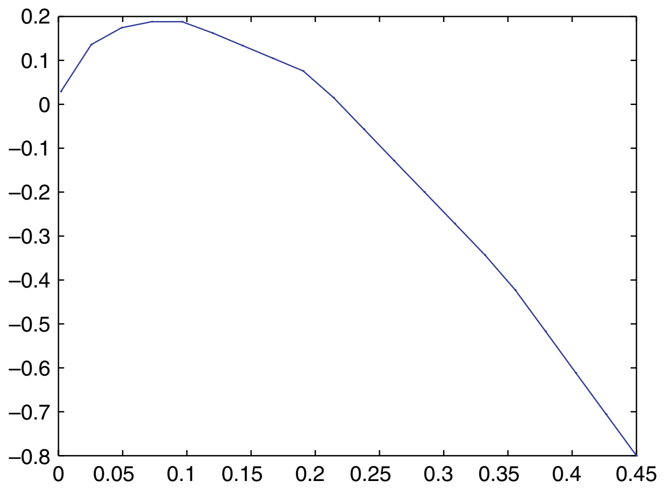

Figure 3.3. *The function $\beta(\lambda)$ is a concave function*

3.4. Optimum transform

In a second step, we find among all the transformations $\mathcal{T}$ the one that minimizes σ_Q^2 after optimum allocation of the bM bits available for the transformed vector. To minimize σ_Q^2 as given by equation [3.8], we need to find the transformation $\mathcal{T}_{\text{opt}}$ that minimizes the geometric mean of the sub-band signal powers. We limit ourselves to the case of orthogonal transforms which already necessitate that $N = M$.

Consider the covariance matrix of the vector $\underline{X}(m)$, which is an $M \times M$-dimensional Toeplitz matrix:

$$\mathbb{R}_X = \sigma_X^2 \begin{bmatrix} 1 & \rho_1 & \cdots & \rho_{M-1} \\ \rho_1 & \ddots & \ddots & \vdots \\ \vdots & \ddots & \ddots & \rho_1 \\ \rho_{M-1} & \cdots & \rho_1 & 1 \end{bmatrix}$$

Writing the covariance matrix of the transformed vector $\underline{Y}(m)$ in the form:

$$\mathbb{R}_Y = \begin{bmatrix} \sigma_{Y_0}^2 & \rho_{0,1}\sigma_{Y_0}\sigma_{Y_1} & \cdots & \rho_{0,M-1}\sigma_{Y_0}\sigma_{Y_{M-1}} \\ \rho_{1,0}\sigma_{Y_1}\sigma_{Y_0} & \ddots & \ddots & \vdots \\ \vdots & \ddots & \ddots & \rho_{M-2,M-1}\sigma_{Y_{M-2}}\sigma_{Y_{M-1}} \\ \rho_{M-1,0}\sigma_{Y_{M-1}}\sigma_{Y_0} & \cdots & \cdots & \sigma_{Y_{M-1}}^2 \end{bmatrix}$$

and setting:

$$\rho_{i,j} = \frac{\mathbb{E}\{Y_i Y_j\}}{\sqrt{\mathbb{E}\{Y_i^2\}\mathbb{E}\{Y_j^2\}}}$$

We find:

$$\mathbb{R}_Y = \begin{bmatrix} \sigma_{Y_0} & 0 & \cdots & 0 \\ 0 & \ddots & \ddots & \vdots \\ \vdots & \ddots & \ddots & 0 \\ 0 & \cdots & 0 & \sigma_{Y_{M-1}} \end{bmatrix} \mathbb{R}_Y^{'} \begin{bmatrix} \sigma_{Y_0} & 0 & \cdots & 0 \\ 0 & \ddots & \ddots & \vdots \\ \vdots & \ddots & \ddots & 0 \\ 0 & \cdots & 0 & \sigma_{Y_{M-1}} \end{bmatrix}$$

with:

$$\mathbb{R}_Y^{'} = \begin{bmatrix} 1 & \rho_{0,1} & \cdots & \rho_{0,M-1} \\ \rho_{1,0} & \ddots & \ddots & \vdots \\ \vdots & \ddots & \ddots & \rho_{M-2,M-1} \\ \rho_{M-1,0} & \cdots & \rho_{M-1,M-2} & 1 \end{bmatrix}$$

The determinant is written as:

$$\det \mathbb{R}_Y = \prod_{k=0}^{M-1} \sigma_{Y_k}^2 \; \det \mathbb{R}_Y^{'} \qquad [3.12]$$

As:

$$\det \mathbb{R}_Y = \det \mathbb{R}_X \det \mathcal{T}\mathcal{T}^t = \det \mathbb{R}_X$$

because the transform is assumed to be orthogonal, we look for the transformation $\mathcal{T}$ that minimizes the geometric mean of the powers, all while keeping the determinant of $\mathbb{R}_Y$ constant. Equation [3.12] shows that the optimum transformation maximizes the determinant of the matrix $\mathbb{R}_Y^{'}$. If the eigenvalues of $\mathbb{R}_Y^{'}$ are $\lambda_0^{'} \cdots \lambda_{M-1}^{'}$, we know [HAY 91] that these eigenvalues are real and non-negative since $\mathbb{R}_Y^{'}$ was defined to be non-negative and the sum of the eigenvalues is equal to M since the trace of a matrix is invariant under orthogonal transformations. We have:

$$\det \mathbb{R}_Y^{'} = \prod_{k=0}^{M-1} \lambda_k^{'} \leq \left(\frac{1}{M} \sum_{k=0}^{M-1} \lambda_k^{'} \right)^M = 1 \qquad [3.13]$$

by applying, once again, the property that the arithmetic mean of a set of real, non-negative numbers is greater than or equal to the geometric mean. Equality is reached when all the eigenvalues are equal to 1, that is, when the components of the transformed vector are decorrelated.

If V is the square matrix constructed from the eigenvectors of $\mathbb{R}_X$ and Λ is the diagonal matrix formed from those eigenvalues, we find:

$$\mathbb{R}_X V = V\Lambda$$

We know that the matrix V is an orthogonal matrix. We can therefore write:

$$V^t \mathbb{E}\{\underline{X}\underline{X}^t\} V = \Lambda$$

$$\mathbb{E}\{V^t \underline{X}(V^t \underline{X})^t\} = \Lambda$$

The components of the vector transformed by $\mathcal{T} = V^t$ are decorrelated. This is therefore the optimum transform and is known as the Karhunen–Loeve transform.

The Karhunen–Loeve transform therefore performs a decomposition of the vector $\underline{X}(m)$ on the eigenvectors of the covariance matrix. It produces decorrelated coefficients and we can show that this is the transform which succeeds in concentrating the highest power in the smallest number of coefficients. However, it has two very important drawbacks. The calculation load is often prohibitive because this transform depends on the signal and calculation algorithm of eigenvectors and eigenvalues of a matrix, based on successive QR factorizations, is very intensive. This transform also has the disadvantage of having no simple frequency interpretation.

3.5. Performance

3.5.1. *Transform gain*

Performing any transform $\underline{y}(m) = \mathcal{T}\underline{x}(m)$ followed by an optimum bit allocation already involves a gain because:

$$G_{\mathrm{t}} = \frac{c(1)\sigma_X^2 2^{-2b}}{c(1)(\prod_{k=0}^{M-1} \sigma_{Y_k}^2)^{1/M} 2^{-2b}} = \frac{1/M \sum_{k=0}^{M-1} \sigma_{Y_k}^2}{(\prod_{k=0}^{M-1} \sigma_{Y_k}^2)^{1/M}}$$

appears as the ratio of the arithmetic and geometric means of positive numbers and can only be greater than or equal to 1. This gain [3], which is due to the transformation and optimum bit allocation, is a function of $\mathcal{T}$ and M.

Furthermore, if we apply the Karhunen–Loeve transform, knowing that we still have:

$$\sigma_X^2 = \frac{1}{M} \sum_{k=0}^{M-1} \lambda_k$$

3. In image coding literature, the term “coding gain” is usually used, which has nothing to do with the coding gain used in channel coding.

where $\lambda_0 \cdots \lambda_{M-1}$ are the eigenvalues of $\mathbb{R}_X$ relative to the normalized eigenvectors, but that:

$$\left(\prod_{k=0}^{M-1} \sigma_{Y_k}^2\right)^{1/M} = \left(\prod_{k=0}^{M-1} \lambda_k\right)^{1/M}$$

only happens in this case, we can note that the transform gain is equal to the ratio of the arithmetic and geometric means of eigenvalues from the covariance matrix $\mathbb{R}_X$:

$$G_{\mathrm{t}}(M) = \frac{1/M \sum_{k=0}^{M-1} \lambda_k}{(\prod_{k=0}^{M-1} \lambda_k)^{1/M}}$$

The Karhunen–Loeve transform results in the highest transform gain.

The geometric mean is written as:

$$\left(\prod_{k=0}^{M-1} \lambda_k\right)^{1/M} = \exp\left(\frac{1}{M}\sum_{k=0}^{M-1} \log_e \lambda_k\right)$$

Therefore, the transform gain is written as:

$$G_{\mathrm{t}}(M) = \frac{(1/M) \sum_{k=0}^{M-1} \lambda_k}{\exp\left((1/M) \sum_{k=0}^{M-1} \log_e \lambda_k\right)}$$

Two results can be shown [GRA 72] :

– If λ_k is an eigenvalue of the covariance matrix $\mathbb{R}_X$ of the process $X(n)$ and if $S_X(f)$ is the power spectral density of this process, then:

$$\lambda_k = S_X(\Omega_k)$$

The M frequencies $\Omega_0 \cdots \Omega_{M-1}$ are known as the eigen frequencies. They do not have the property of being uniformly distributed in the interval $[0, 1]$.

– Whatever the function $g(.)$ may be, we have:

$$\lim_{M\to\infty} \frac{1}{M}\sum_{k=0}^{M-1} g(\lambda_k) = \int_{-1/2}^{1/2} g[S_X(f)]\mathrm{d}f$$

This result is known as Szego's theorem.

From these two results, we can deduce that:

$$G_{\mathrm{t}}(\infty) = \frac{\int_{-1/2}^{1/2} S_X(f)\mathrm{d}f}{\exp(\int_{-1/2}^{1/2} \log_e[S_X(f)]\mathrm{d}f)}$$

We recognize the expression for the prediction gain given by equation [1.13]. We therefore have:

$$G_{\mathrm{t}}(\infty) = G_{\mathrm{p}}(\infty)$$

Asymptotically, the transform coding and predictive quantization techniques give identical results. However, for a given value of the parameter M, predictive quantization of order M can appear preferable to using an optimum M-dimensional transform since the transform quantization makes use of the correlation contained in the signal up to order M. However, predictive quantization makes use of higher-order correlation thanks to the prediction loop. If the process undergoing quantization is auto-regressive of order P, we need only to realize a prediction of order P so that

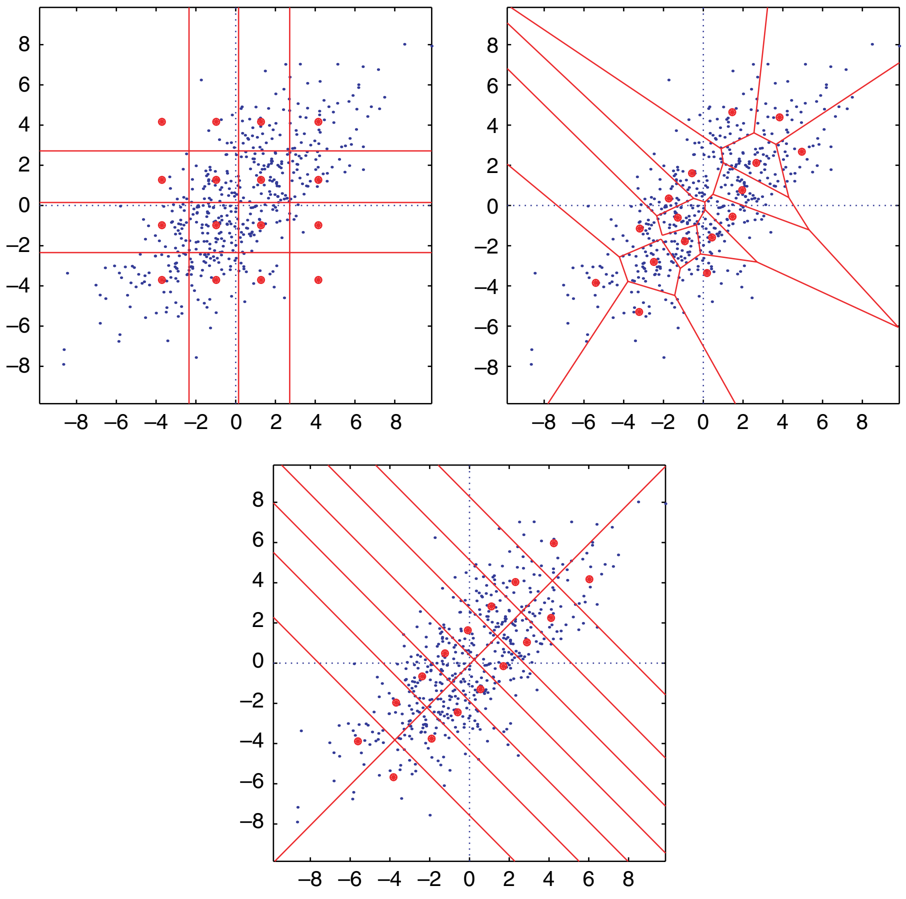

Figure 3.4. *Different quantization methods when $b = 2$*

the prediction gain reaches its asymptotic value. This is not the case for the transform quantizer.

We can note that using the Karhunen–Loeve transform involves making sure that $\{\sigma^2_{Y_0} \cdots \sigma^2_{Y_{M-1}}\}$ is the most faithful estimate possible of $S_X(f)$.

3.5.2. *Simulation results*

The signal shown in Figure 2.1 is quantized at a resolution of 2 bits per sample. Figure 3.4 shows the partitions and reproduction values which correspond to a non-uniform scalar quantization, a vector quantization and a scalar quantization with optimum bit allocation preceded by a Karhunen–Loeve transform (for $N = M = 2$, this means a straightforward 45 degree rotation of the axes).

Chapter 4

Entropy Coding

4.1. Introduction

Consider a continuous-time signal $x(t)$ bandlimited in the range $[-B, +B]$. The signal is sampled at a frequency greater than or equal to the Nyquist frequency, $f_e = 2B$. Using this method, a discrete-time signal $x(n)$ is obtained. This signal may be interpreted as the realization of a discrete-time random process $X(n)$. The random process is assumed to have the common stationarity and ergodicity properties.

This process has continuous values. Assume that it has been quantized with a significant resolution. This random process becomes a discrete-valued random process, that is, $X(n)$ takes values from a finite set. In information theory, $X(n)$ is the information source, the finite set $A_X = \{x^1 \cdots x^{L_X}\}$ is the input alphabet, and the elements x^i are the input symbols or letters of the alphabet. Next, we aim to compress this information. The aim of this chapter is to explain that, with certain hypotheses, carrying out this operation is possible without introducing any distortion. This is known as noiseless or lossless coding or entropy coding. Unfortunately, the allowed compression rates are generally too low. Therefore, a certain level of distortion is tolerated. We can show that a function, known as the rate–distortion function, gives a lower limit for distortion when the bit rate is set or, inversely, a lower limit for the bit rate when the distortion level is set. In this chapter, we also introduce the notion of a channel's transmission capacity in order to set out the combined source-channel coding theorem. We will re-examine the scalar quantization procedure and introduce a new quantization method – scalar quantization with entropy constraints – which we compare to the quantizer examined in the previous chapters and the Lloyd–Max quantizer.

The presentation in this chapter is rather brief. The majority of theorems are not shown. The reader is left to consult the works of Shannon [SHA 48, SHA 59], which form the basis for information theory and reference works [GAL 68, BER 71, BLA 87, GRA 90] and especially [COV 91]. But that presentation is also incomplete. For example, he does not describe the Ziv–Lempel algorithm [COV 91], which is widely used to compress information through the UNIX command *compress*.

4.2. Noiseless coding of discrete, memoryless sources

Assume, in the first instance, that the process $X(n)$ is a sequence of independent and identically distributed variables (i.i.d.), with values in the range $A_X = \{x^1 \cdots x^{L_X}\}$. This is a discrete, memoryless source. Note that:

$$p_X(i) = Prob\{X(n) = x^i\}$$

is the distribution of different probabilities. This distribution is independent of the observation time n since the random process is time-invariant.

4.2.1. *Entropy of a source*

The self-information of the event $\{X(n) = x^i\}$ is:

$$I(x^i) = -\log_2 p_X(i)$$

This quantity is positive or zero and expressed in bits/symbol[1] since a logarithm with base 2 has been taken. The entropy of the source $X(n)$ is the expected value (mathematical expectation) of the random variable $I(x^i)$:

$$H(X) = E\{I(x^i)\}$$

$$H(X) = -\sum_{i=1}^{L_X} p_X(i) \log_2 p_X(i) \qquad [4.1]$$

The entropy[2] is the amount of information that a realization of $X(n)$ carries on average. It gives the numbers of bits necessary, on average, to completely write the sources. It measures the uncertainty associated with the source.

1. If following the UIT-T recommendations, use units of Shannon/symbol to justifiably avoid confusion with the bit which is the unit of counting binary symbols used in coding information. The common terminology is used here.
2. The usual notation $H(X)$ does not mean that H is a function of values taken by the random variable X. Entropy depends only on the probability distribution $p_X(i)$.

Take the example of a binary source, $X(n) \in \{x^1, x^2\}$, without memory and written $p = p_X(1)$ to simplify the notation. The entropy of this source is:

$$H(X) = -p \log_2 p - (1-p) \log_2(1-p)$$

This is illustrated in Figure 4.1. The entropy is positive or zero. The entropy is zero if $p = 0$ or $p = 1$, that is, if the source is totally predictable. The entropy is highest and equal to 1 if the two symbols are equiprobable. This is a concave function of p.

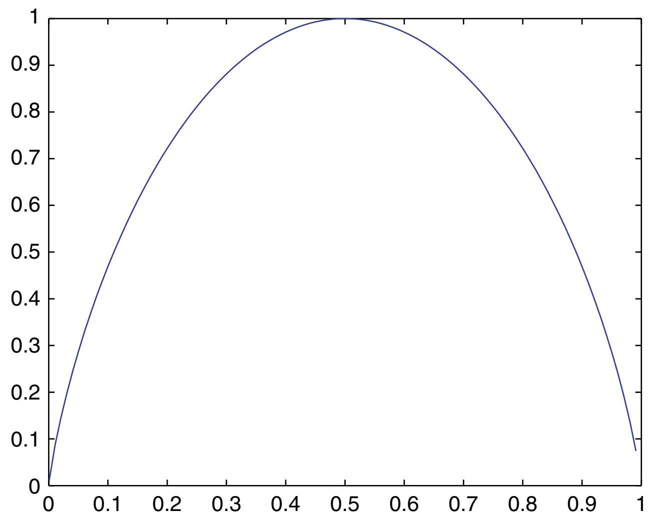

Figure 4.1. *Entropy of a binary, memoryless source as a function of the probability p of obtaining one of two symbols*

The approach proposed by Shannon in his fundamental paper [SHA 48] to justify this definition can be schematically summed up as follows. Consider the binary source, $X(n) \in \{x^1, x^2\}$, assumed to be memoryless and observed at N times. A random generator produces a particular realization, for example $[x^2 x^1 x^1 x^2 \cdots x^2]$. If N_1 is the number of times that x^1 appears and N_2 the number of times that x^2 appears, the (weak) law of large numbers dictates that if N is large, $N_1/N \approx p$ and $N_2/N \approx 1-p$. Since $X(n)$ is a set of independent random variables, the probability of obtaining the observed realization is:

$$\text{Prob}\{\underline{X} = [x^2 x^1 x^1 x^2 \cdots x^2]\} = p^{N_1}(1-p)^{N_2} = p^{Np}(1-p)^{N(1-p)}$$

As this probability is approximately constant (a property known in the literature as the *asymptotic equirepartition property*), the number of different possible realizations (known as *typical sequences*) is given by:

$$N_{st} = \left(p^{Np}(1-p)^{N(1-p)}\right)^{-1}$$

As:

$$\log_2 p^{-Np}(1-p)^{-N(1-p)} = -Np\log_2 p - N(1-p)\log_2(1-p) = NH(X)$$

and the total number of sequences is L_X^N, we find:

$$N_{st} = 2^{NH(X)} \leq L_X^N = 2^{N\log_2(L_X)}$$

This simply means that, amidst all possible sequences, a certain number has a probability of zero. It is useless to allow bits for coding sequences which are never realized!

All of the results presented for $L_X = 2$ can be generalized when $L_X > 2$. Equation [4.1] shows that the entropy of a discrete source is either positive or zero. The entropy is zero when $X(n)$ is almost certainly equal to a particular value x^i; therefore, the source is totally predictable. The entropy is at a maximum value of $\log_2 L_X$ when the symbols are equiprobable. We have:

$$0 \leq H(X) \leq \log_2 L_X$$

4.2.2. *Coding a source*

4.2.2.1. *Definitions*

In the previous chapters, the word coding has not been precisely defined. We have simply said that it involves associating a number, $i(n)$ or $i(m) \in \{1 \cdots L\}$, where $L = 2^{bN}$ and the parameter b specifies the number of bits per sample, with a scalar $x(n)$ or a vector $\underline{x}(m) = [x(mN) \cdots x(mN+N-1)]^t$. In this chapter, the information to transmit or store, modeled by the random process $X(n)$, takes its values from a finite set, the input alphabet A_X, and we aim to represent (encode) the different elements of this set in a way that is both adapted to the transmission channel characteristics and efficient.

Adapted to the transmission channel characteristics means that the representation of each input symbol, a code word, can be constructed from elements of another alphabet which is adapted to the channel. From here onward, assume that this alphabet is composed of two elements $A_C = \{a^1, a^2\}$, for example, the two usual binary symbols 0 and 1.

In an efficient manner means that we aim to represent the source by using the least number of bits, that is, minimizing the average length of the code words. Up until now, all code words have had the same length, bN. From now on, we will associate the most probable input symbol with the shortest code words. The code is said to be of variable length.

More exactly, the coding of the source $X(n)$ is a mapping of the alphabet A_X in the set of finite sequences of elements from the alphabet A_C. The code $C = \{c^1 \cdots c^{L_X}\}$ is the set of these sequences. Each possible sequence $c^i = [a^{i(1)} \cdots a^{i(l)}]$ is a code word. The number of elements of A_C in a word is the length l of the word. The average length of the code words is given by:

$$\bar{l} = \sum_{i=1}^{L_X} p_X(i) l(c^i)$$

4.2.2.2. *Uniquely decodable instantaneous code*

We can immediately notice that there appears to be a problem with a variable length code: are separators needed between the code words in a sequence? They are not needed when the code satisfies what is known as the prefix condition: no code word may be a prefix of any other code word.

Take the example of a source which can only take six different values $\{x^1 \cdots x^6\}$. The code defined by the associations given in Table 4.1 is a valid code, uniquely decodable, since when a binary stream is received, for example 11001110101110, we can deduce the sequence x^4, x^1, x^5, x^3, x^4. From this, it can be seen that the symbol decoding can be done without reference to future code words. The code is called instantaneous. In the following sections, we limit ourselves to this particular case.

Symbols	x^1	x^2	x^3	x^4	x^5	x^6
Codes	0	100	101	110	1110	1111

Table 4.1. *A code definition*

The previous code also satisfies the prefix condition. One simple method of satisfying the prefix condition or to construct a code which satisfies this condition is to draw an oriented graph in the shape of a binary tree, label each branch from a node with the symbol 0 or 1, and associate a code word with each terminal node by taking the sequence of binary symbols on the branches as the code word, as shown in Figure 4.2. This tree is not quite complete. The number of branches between the initial node, the root, and the terminal node, a leaf, is the length of the associated code word.

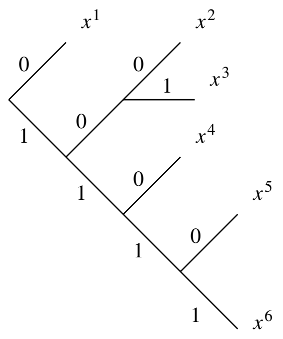

Figure 4.2. *Binary tree associated with a uniquely decodable code*

4.2.2.3. *Kraft inequality*

Consider a binary code $C = \{c^1 \cdots c^{L_X}\}$ which represents a source without errors. The necessary and sufficient condition for the code to satisfy the prefix condition is:

$$\sum_{i=1}^{L_X} 2^{-l(c^i)} \leq 1 \qquad [4.2]$$

where $l(c^i)$ is the length of the code word c^i. For example, in Figure 4.2, we find:

$$2^{-1} + 3 \times 2^{-3} + 2 \times 2^{-4} = 1$$

This simply demonstrates the key principle of the necessary condition when L_X is a power of 2. Consider the complete tree which represents the set of the L_X words of a code $C^{'}$ in which all the code words are of the same length $l_{\max}$ such as $L_X = 2^{l_{\max}}$. This code satisfies relation [4.2] since:

$$\sum_{i=1}^{L_X} 2^{-l_{\max}} = 2^{l_{\max}} 2^{-l_{\max}} = 1$$

We can move from the tree for the code C' to the tree for the code C by pruning a certain number of branches and creating new ones from them. Pruning the branches does not modify the first part of [4.2] since we always replace two terms in the form $2^{-l} \times 2^{-l+1}$. This will even be true if new branches are created since 2^{-l} is replaced by $2 \times 2^{-l-1}$.

4.2.2.4. *Optimal code*

The constraint that the quantities $l(c^i)$ must be integers is relaxed, and the inequality symbol is replaced by an equality in [4.2]. The optimal code is the one which minimizes the average length:

$$\bar{l} = \sum_{i=1}^{L_X} p_X(i) l(c^i)$$

within the constraint:

$$\sum_{i=1}^{L_X} 2^{-l(c^i)} = 1$$

We can introduce a Lagrange multiplier for $i = 1 \cdots L_X$:

$$\frac{\partial}{\partial l(c^i)} \left[\sum_{j=1}^{L_X} p_X(j) l(c^j) + \lambda \sum_{j=1}^{L_X} 2^{-l(c^j)} \right] = 0$$

We find:

$$p_X(i) - \lambda \log_e 2\, 2^{-l(c^i)} = 0$$

such that:

$$2^{-l(c^i)} = \frac{p_X(i)}{\lambda \log_e 2}$$

Since:

$$\sum_{i=1}^{L_X} 2^{-l(c^i)} = \frac{1}{\lambda \log_e 2} \sum_{i=1}^{L_X} p_X(i) = 1$$

the value of the constant is set as $\lambda \log_e 2 = 1$. We find:

$$2^{-l(c^i)} = p_X(i)$$

$$l(c^i) = -\log_2 p_X(i)$$

The mean length which corresponds to the optimal code is given by:

$$\bar{l} = \sum_{i=1}^{L_X} p_X(i) l(c^i) = -\sum_{i=1}^{L_X} p_X(i) \log_2 p_X(i)$$

$$\bar{l} = H(X)$$

Among all the codes which satisfy the prefix condition, the one which minimizes the average code word length has a mean length equal to the source entropy. The entropy $H(X)$ therefore appears to be a fundamental limit in representing an information source without distortion.

4.2.3. *Theorem of noiseless coding of a memoryless discrete source*

The following explanation aims quite simply to make the preceding result more precise when the code word lengths are integers. These results were demonstrated by Shannon in 1948 [SHA 48].

4.2.3.1. *Proposition 1*

If $p(1) \cdots p(L)$ and $q(1) \cdots q(L)$ are two ordinary probability distributions:

$$\sum_{i=1}^{L} p(i) \log_2 \frac{q(i)}{p(i)} \leq 0$$

The equality is true if and only if $p(i) = q(i)$ for any $i = 1 \cdots L$.

In effect, we know that for all positive x:

$$\log_2 x \leq (x-1) \log_2 e$$

Therefore:

$$\log_2 \frac{q(i)}{p(i)} \leq \left(\frac{q(i)}{p(i)} - 1 \right) \log_2 e$$

or:

$$\sum_{i=1}^{L} p(i) \log_2 \frac{q(i)}{p(i)} \leq \sum_{i=1}^{L} p(i) \left(\frac{q(i)}{p(i)} - 1 \right) \log_2 e = 0$$

Equality is achieved if and only if $q(i)/p(i) = 1$ for any i.

The expression:

$$D(p||q) = \sum_{i=1}^{L} p(i) \log_2 \frac{p(i)}{q(i)} \qquad [4.3]$$

is known as the relative entropy or Kullback–Leibler distance between the two probability distributions. It is always positive or zero. It can be interpreted as a measure of the distance between two probability distributions although this is not, strictly speaking, a distance since it is not a symmetrical expression and it does not respect the triangular inequality.

4.2.3.2. *Proposition 2*

All encoding of the source $X(n)$ by a uniquely decodable instantaneous code has a mean length which satisfies:

$$H(X) \leq \bar{l}$$

In effect, all uniquely decodable instantaneous code satisfies the Kraft inequality:

$$\sum_{i=1}^{L_X} 2^{-l(c^i)} \leq 1$$

A new probability distribution is created in the form:

$$q(i) = a2^{-l(c^i)}$$

where $a \geq 1$ since:

$$\sum_{i=1}^{L_X} q(i) = 1 = a \sum_{i=1}^{L_X} 2^{-l(c^i)}$$

Proposition 1 says that:

$$\sum_{i=1}^{L_X} p_X(i) \log_2 a \frac{2^{-l(c^i)}}{p_X(i)} \leq 0$$

such that:

$$-\sum_{i=1}^{L_X} p_X(i) \log_2 p_X(i) - \sum_{i=1}^{L_X} p_X(i) l(c^i) + \log_2 a \leq 0$$

$$H(X) \leq \bar{l} - \log_2 a$$

Hence, since $\log_2 a \geq 0$, the desired equation is obtained.

4.2.3.3. *Proposition 3*

A uniquely decodable instantaneous code exists which satisfies:

$$\bar{l} < H(X) + 1$$

In effect, we choose a code such that:

$$-\log_2 p_X(i) \leq l(c^i) < -\log_2 p_X(i) + 1$$

From the first inequality:

$$-l(c^i) \leq \log_2 p_X(i)$$

we find:

$$2^{-l(c^i)} \leq p_X(i)$$

$$\sum_{i=1}^{L_X} 2^{-l(c^i)} \leq \sum_{i=1}^{L_X} p_X(i) = 1$$

From this the existence of a code having this length distribution is deduced. From the second inequality, we find:

$$\sum_{i=1}^{L_X} p_X(i) l(c^i) < -\sum_{i=1}^{L_X} p_X(i) \log_2 p_X(i) + 1$$

$$\bar{l} < H(X) + 1$$

4.2.3.4. *Theorem*

For all memoryless discrete sources $X(n)$, there exists a code which instantaneously represents the source exactly and is uniquely decodable, satisfying:

$$H(X) \leq \bar{l} < H(X) + 1 \qquad [4.4]$$

where $H(X)$ is the source entropy and $\bar{l}$ is the mean code length.

4.2.4. *Constructing a code*

4.2.4.1. *Shannon code*

The most straightforward procedure is to choose:

$$l(c^i) = \lceil -\log_2 p_X(i) \rceil$$

where $\lceil x \rceil$ represents the smallest integer greater than or equal to x. We have:

$$\bar{l} = \sum_{i=1}^{L_X} p_X(i) \lceil -\log_2 p_X(i) \rceil$$

$$\bar{l} \leq -\sum_{i=1}^{L_X} p_X(i) \log_2 p_X(i) + \sum_{i=1}^{L_X} p_X(i)$$

$$\bar{l} \leq H(X) + 1$$

Since:

$$2^{-\lceil -\log_2 p_X(i) \rceil} \leq 2^{\log_2 p_X(i)}$$

$$\sum_{i=1}^{L_X} 2^{-l(c^i)} \leq \sum_{i=1}^{L_X} p_X(i)$$

$$\sum_{i=1}^{L_X} 2^{-l(c^i)} \leq 1$$

from this we deduce that an instantaneous code exists which satisfies the prefix condition.

4.2.4.2. *Huffman algorithm*

The algorithm proposed by Huffman in 1951 involves the progressive construction of a binary tree starting with the terminal nodes:

– start from the two lists $\{x^1 \cdots x^{L_X}\}$ and $\{p_X(1) \cdots p_X(L_X)\}$;

– select the two least probable symbols, create two tree branches, and label them with the binary symbols 0 and 1;

– the two lists are updated by bringing together the two symbols used previously and a new symbol, which is given a probability which is the sum of the probabilities for the two previously selected symbols;

– the two preceding steps are repeated until no more symbols remain in the list.

We can show that this algorithm is the optimal one. Any other uniquely decodable code will have a lower mean word length to code produced via this algorithm.

4.2.4.3. *Example 1*

Take the example of a source which can only take six different values and assume that their probabilities are known. These are given in Table 4.2. The entropy of this source is 2.06 bits. The Shannon code has a mean length of 2.28 bits.

Symbols	x^1	x^2	x^3	x^4	x^5	x^6
Probabilities	0.5	0.15	0.17	0.08	0.06	0.04

Table 4.2. *Probabilities associated with six events* $\{X(n) = x^i\}$

The Huffman algorithm provides the binary tree shown in Figure 4.3. The coding table is presented in Table 4.1. The mean length is 2.1 bits, very close to the theoretical limit.

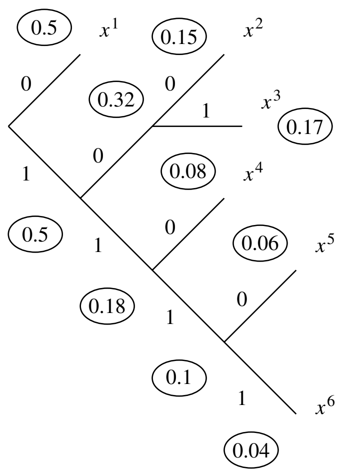

Figure 4.3. *Diagram of the Huffman algorithm*

4.2.5. *Generalization*

The precision of the double inequality [4.4] is too low since the value of $H(X)$ is generally low. To improve the precision, a random vector, $\underline{X}_N$, is created by regrouping N random variables $X(mN) \cdots X(mN+N-1)$ and a code word is associated with all possible realizations of this vector. It corresponds, in this case, to an application of the product $A_X \times \cdots \times A_X$ in the set of finite sequences. The memoryless source hypothesis is maintained in this section.

4.2.5.1. *Theorem*

We can show that if we regroup N symbols from the source and associate with them a code word c^i of length $l(c^i)$, there exists a code such that the mean length:

$$\bar{l} = \sum_{\underline{X}_N} Pr(\underline{X}_N) l(c^i)$$

satisfies:

$$H(X) \leq \frac{\bar{l}}{N} < H(X) + \frac{1}{N} \qquad [4.5]$$

The ratio $\bar{l}/N$ represents the mean number of bits per symbol that is necessary and sufficient for representing the source exactly.

Another theorem exists which assumes that the lengths of all code words are identical and allows N to vary. If we note p_e as the probability that the sequence $X(mN) \cdots X(mN + N - 1)$ cannot be associated with a code word, this theorem says that p_e can be made arbitrarily small, provided that N is sufficiently large.

4.2.5.2. *Example 2*

Assume that a memoryless source can only take two values with known probabilities. Let us create a two-dimensional vector and a three-dimensional vector. The set of probabilities for different events is given in Table 4.3. The entropy of this source is 0.88 bit.

			x^1	x^2			
			0.7	0.3			
		x^1x^1	x^1x^2	x^2x^1	x^2x^2		
		0.49	0.21	0.21	0.09		
$x^1x^1x^1$	$x^1x^1x^2$	$x^1x^2x^1$	$x^1x^2x^2$	$x^2x^1x^1$	$x^2x^1x^2$	$x^2x^2x^1$	$x^2x^2x^2$
0.343	0.147	0.147	0.063	0.147	0.063	0.063	0.027

Table 4.3. *Probabilities of 2, 4, and 8 possible events*

The mean numbers of bits per symbol are given in Table 4.4. They do not create a decreasing sequence when N increases but they do satisfy the double inequality [4.5].

N	1	2	3
l/N	1	0.905	0.908

Table 4.4. *Mean numbers of bits per symbol*

4.2.6. ***Arithmetic coding***

There is no longer the question of finding an instantaneous code as above. This coding technique encodes several symbols at the same time. Consider five symbols which are assumed to have the probability distribution shown in Table 4.5 and associate part of the interval $[0-1[$ with each symbol as a function of its probability as shown in the table.

To encode (simultaneously), for example, the three symbols $[x^3, x^2, x^4]$ requires finding a sub-set of the interval $[0-1[$ as follows:

– first of all, select the interval $[0.3-0.4[$ associated with the symbol x^3;

Symbols	x^1	x^2	x^3	x^4	x^5
Probabilities	0.1	0.2	0.1	0.4	0.2
Intervals	[0–0.1[	[0.1–0.3[	[0.3–0.4[	[0.4–0.8[	[0.8–1[

Table 4.5. *Probabilities associated with five events $\{X(n) = x^i\}$ and coding interval definition*

– next, "include" the interval $[0.1-0.3[$ specified for the symbol x^2 in the interval $[0.3-0.4[$ to give the interval $[0.31-0.33[$;

– continue with x^4 to find the limits of the new interval:

$$0.31 + 0.4 \times (0.33 - 0.31) = 0.318$$

$$0.31 + 0.8 \times (0.33 - 0.31) = 0.326$$

Encoding $[x^3, x^2, x^4]$ amounts to transmitting whatever real number belonging to the interval $[0.318-0.326[$, for example 0.32, or more exactly the binary representation which is the most economical possible, for example $\{0101001\}$ since $2^{-2}+2^{-4}+2^{-7}$ = 0.3203.

At the decoder, the following processing is carried out:

– since 0.3203 belongs to the interval $[0.3-0.4[$, we can determine x^3;

– since:

$$\frac{0.3203 - 0.3}{0.4 - 0.3} = 0.202 \in [0.1-0.3[$$

we determine x^2;

– etc.

For further details, especially on how the most economical binary representations are obtained in practice, the reader is referred to the article [HOW 94].

4.3. Noiseless coding of a discrete source with memory

The presentation above examined the most elementary case. In general, a statistical dependence exists between the successive samples from the source which can be exploited to reduce the number of bits necessary to represent the source exactly.

4.3.1. *New definitions*

Consider two discrete random variables[3] X and Y taking values from $A_X = \{x^1 \cdots x^{L_X}\}$ and $A_Y = \{y^1 \cdots y^{L_Y}\}$, respectively, with joint probabilities:

$$p_{XY}(i,j) = \text{Prob}\{X = x^i, Y = y^j\}$$

and conditional probabilities:

$$p_{Y|X}(i|j) = \text{Prob}\{Y = y^j | X = x^i\}$$

We have:

– the joint information of the two events $\{X = x^i\}$ and $\{Y = y^j\}$:

$$I(x^i, y^j) = -\log_2 p_{XY}(i,j)$$

– the joint entropy of two discrete random variables X and Y, the expectation of the joint information:

$$H(X,Y) = -\sum_{i=1}^{L_X} \sum_{j=1}^{L_Y} p_{XY}(i,j) \log_2 p_{XY}(i,j)$$

– the conditional information of the event $\{Y = y^j\}$, knowing that the event $\{X = x^i\}$ has happened:

$$I(y^j | x^i) = -\log_2 p_{Y|X}(j|i)$$

– the conditional entropy of the discrete random variable Y, knowing that event $\{X = x^i\}$ has happened:

$$H(Y|x^i) = -\sum_{j=1}^{L_Y} p_{Y|X}(j|i) \log_2 p_{Y|X}(j|i)$$

– the conditional entropy of Y given X, the expectation of the preceding conditional entropy, that is:

$$H(Y|X) = \sum_{i=1}^{L_X} p_X(i) H(Y|x^i)$$

$$H(Y|X) = -\sum_{i=1}^{L_X} \sum_{j=1}^{L_Y} p_{XY}(i,j) \log_2 p_{Y|X}(j|i)$$

3. For example, $X(n)$ and $X(n+k)$. In this case, $L_X = L_Y$.

The relation:

$$p_{XY}(i,j) = p_X(i)p_{Y|X}(j|i)$$

leads to the following relation between the entropy $H(X)$, the joint entropy $H(X,Y)$, and the conditional entropy $H(Y|X)$:

$$H(X,Y) = H(X) + H(Y|X) \qquad [4.6]$$

In effect:

$$H(X,Y) = -\sum_{i=1}^{L_X}\sum_{j=1}^{L_Y} p_{XY}(i,j)\log_2 p_{XY}(i,j)$$

$$H(X,Y) = -\sum_{i=1}^{L_X}\sum_{j=1}^{L_Y} p_{XY}(i,j)\log_2 p_X(i)p_{Y|X}(j|i)$$

$$H(X,Y) = -\sum_{i=1}^{L_X} p_X(i)\log_2 p_X(i) - \sum_{i=1}^{L_X}\sum_{j=1}^{L_Y} p_{XY}(i,j)\log_2 p_{Y|X}(j|i)$$

4.3.2. *Theorem of noiseless coding of a discrete source with memory*

The above definitions and relation [4.6] can be generalized to a case of N discrete random variables. Consider the vector:

$$\underline{X}_N = [X(mN)\cdots X(mN+N-1)]^t$$

and the joint probability:

$$\text{Prob}\{X(mN) = x^{i_1}\cdots X(mN+N-1) = x^{i_N}\} = p_{\underline{X}_N}(i_1\cdots i_N) = p_{\underline{X}_N}(\underline{i})$$

The entropy of this vector is defined by:

$$H(\underline{X}_N) = -\sum_{\underline{i}} p_{\underline{X}_N}(\underline{i})\log_2 p_{\underline{X}_N}(\underline{i})$$

The entropy rate is:

$$\bar{H}(X) = \lim_{N\to\infty}\frac{1}{N}H(\underline{X}_N)$$

We can show that a uniquely decodable code can be associated with a source as long as the mean number of bits per symbol is greater than or equal to the entropy rate.

If the source is memoryless, we have:

$$H(\underline{X}_N) = -N \sum_i p_X(i) \log_2 p_X(i) = NH(X)$$

This source's entropy rate is therefore equal to the source entropy. More generally, we have the inequality:

$$0 \leq \bar{H}(X) \leq H(X) \leq \log_2 L_X$$

This whole presentation shows that a coding exists which is capable of exactly representing the source at a bit rate equal to the entropy rate, but has not yet shown how to construct such a code.

We can also note that an efficient coding can only be realized dependent on the reconstruction delay.

4.3.3. *Example of a Markov source*

4.3.3.1. *General details*

Consider the case of a discrete source $X(n)$ which takes its values from the set $\{x^1 \cdots x^{L_X}\}$ and assume that all conditional probabilities are known for an event $\{X(n) = x^i\}$, knowing that $\{X(n-1) = x^j\}$ occurred immediately beforehand. We assume that the conditional probabilities are independent at the time of observation. This is known as a Markov source of order 1. We write:

$$p(i|j) = p_{X(n)|X(n-1)}(i|j) = Prob\{X(n) = x^i | X(n-1) = x^j\}$$

This source can be represented as a state diagram, as shown in Figure 4.4, where the different events $\{X(n) = x^i\}$ are the possible states and the conditional probabilities $p(i|j)$ are the transition probabilities.

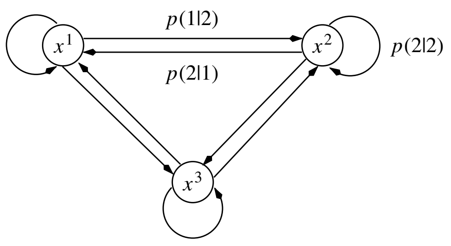

Figure 4.4. *State diagram of a Markov source*

We write that the probability of being in state x^i is:

$$p(i) = Prob\{X(n) = x^i\}$$

Assume that $p(i)$ is independent of the time of observation n. Therefore, we are not concerned with a possible transitory regime.

If a probability $p(i|i)$ is significant, this means that the symbol x^i will, in all likelihood, appear several times in a row. Therefore coding this symbol by specifying its number of repeats is of interest. This is known as run-length coding.

4.3.3.2. *Example of transmitting documents by fax*

This example concerns transmitting and reproducing a typed, black-and-white A4 document ($210 \times 297 \ \mathrm{mm}^2$) in approximately one minute at 4.8 kbit/s [CCI 80] via the public telephone network. In this section, various pieces of information are given which enable us to understand the encoding method used. The description here is limited to one-dimensional encoding specified in the CCITT recommendation. A second encoding method, two-dimensional encoding, exists which takes account of information relative to the previous line, but this method is too specific for it to be of interest in this work.

The A4 page is read from left to right and from top to bottom. A definition of 4 lines per mm is used with 1728 pixels in black and white per line. There are therefore around 2 Mbits of data to transmit. In one minute at 4.8 kbit/s, around 300 kbits are transmitted. The compression rate must therefore be around 7.

In this case, the source is $X(n)$, the sequence of pixels is defined by scanning across the lines, and we can assume that it is a Markov source of order 1. This source can be in one of two states, x^1 and x^2, which correspond to black and white pixels respectively. Note that all probabilities can be completely determined through knowing the two probabilities $p(2|1)$ and $p(1|2)$ since:

$$\begin{aligned} p(1|1) &= 1 - p(2|1) \\ p(2|2) &= 1 - p(1|2) \end{aligned}$$

(it has to go somewhere) and that:

$$\begin{aligned} p(1) &= [1 - p(2|1)]p(1) + p(1|2)p(2) \\ p(2) &= 1 - p(1) \end{aligned}$$

such that:

$$\begin{aligned} p(1) &= \frac{p(1|2)}{p(2|1)+p(1|2)} \\ p(2) &= \frac{p(2|1)}{p(2|1)+p(1|2)} \end{aligned}$$

If the source is in state x^1, the conditional entropy $H(X(n)|X(n-1) = x^1)$ represents the number of bits necessary to encode its evolution from state x^1. We have:

$$H(X(n)|X(n-1) = x^1) = -p(1|1)\log_2 p(1|1) - p(2|1)\log_2 p(2|1) = f(p(2|1))$$
$$f(p(2|1)) = f(u)$$

where $u = p(2|1)$; $f(u) = -u\log_2 u - (1-u)\log_2(1-u)$ for $0 \leq u \leq 1$ which is shown as a graph in Figure 4.1. Further, we find:

$$H(X(n)|X(n-1) = x^2) = f(p(1|2))$$

The entropy rate is the mean (weighted by the probability density of being in the corresponding state) of the conditional entropies. We find:

$$H(X(n)|X(n-1)) = p(1)H(X(n)|X(n-1) = x^1) + p(2)H(X(n)|X(n-1) = x^2)$$
$$H(X(n)|X(n-1)) = g(p(1|2), p(2|1))$$

In conclusion, without knowledge of the source characteristics, 1 bit must be used per symbol. However, if the transition probabilities are low, the entropy rate is low and an efficient encoding must exist.

To construct the code, the most straightforward method is to:

– scan a large number of sheets of writing, drawings, etc. to create training data;

– produce two histograms of the number of blacks (and whites) consecutively so that the probabilities of various events can be estimated: a single black (white), two blacks (whites), etc., k consecutive blacks (whites);

– apply the Huffman algorithm;

– compare the efficiency of this method with the entropy rate calculated from an estimate of $p(1|2)$ and $p(2|1)$.

The recommendation published in 1980 stipulates that each code word represents the number of consecutive white or black pixels. This is therefore a run-length coding. The sequences of black and white are produced alternately. All lines begin with a

0	00110101
1	000111
2	0111
3	1000
4	1011
5	1100
6	1110
7	1111
8	10011
9	10100
10	00111
11	01000
12	001000
13	000011
14	110100
15	110101
16	101010
17	101011
18	0100111
19	0001100
20	0001000
21	0010111
22	0000011
23	0000100
24	0101000
25	0101011
26	0010011
27	0100100
28	0011000
29	00000010
30	00000011
31	00011010

32	00011011
33	00010010
34	00010011
35	00010100
36	00010101
37	00010110
38	00010111
39	00101000
40	00101001
41	00101010
42	00101011
43	00101100
44	00101101
45	00000100
46	00000101
47	00001010
48	00001011
49	01010010
50	01010011
51	01010100
52	01010101
53	00100100
54	00100101
55	01011000
56	01011001
57	01011010
58	01011011
59	01001010
60	01001011
61	00110010
62	00110011
63	00110100

64	11011
128	10010
192	010111
256	0110111
320	00110110
384	00110111
448	01100100
512	01100101
576	01101000
640	01100111
704	011001100
768	011001101
832	011010010
896	011010011
960	011010100
1024	011010101
1088	011010110
1152	011010111
1216	011011000
1280	011011001
1344	011011010
1408	011011011
1472	010011000
1536	010011001
1600	010011010
1664	011000
1728	010011011
EOL	000000000001

Table 4.6. *Code words which specify the length of white runs in fax transmission of documents.*

sequence of white, which can be of length 0, and ends with a special code word (EOL). The code words vary in length. There are two lists of distinct codes for the sequences corresponding to white and black. The sequences of 0–63 white pixels are given code words as shown in Table 4.6.

If a run length is greater than or equal to 64 pixels, a first code word is used to specify which interval it belongs to and a second code word specifies the remainder of the division by 64. There is a second table of code words which specify the lengths of black pixel runs (not included in this book).

4.4. Scalar quantizer with entropy constraint

4.4.1. *Introduction*

At the beginning of Chapter 1, a highly intuitive presentation of the uniform scalar quantizer was given, then we addressed the issue of determining the optimum scalar quantizer for a process $X(n)$ which has an ordinary marginal probability density function $p_X(x)$. Recall that the equation for the quantization error power is:

$$\sigma_Q^2 = \sum_{k=1}^{L} \int_{x \in \Theta^k} (x - \hat{x}^k)^2 p_X(x) dx$$

as a function of the partition $\{\Theta^1 \cdots \Theta^L\}$ of the real axis with reproduction values $\{\hat{x}^1 \cdots \hat{x}^L\}$ when the quadratic error is chosen as the distortion measure. Minimizing σ_Q^2 relative to the parameters t^k which defined the boundary between partitions Θ^k and Θ^{k+1} and relative to the reproduction values $\hat{x}^k$ does not provide a directly usable analytical solution. We simply find two necessary conditions for optimization:

$$t^k = \frac{\hat{x}^k + \hat{x}^{k+1}}{2}$$

$$\hat{x}^k = \mathbb{E}\{X | X \in \Theta^k\}$$

These two conditions are used in the Lloyd–Max algorithm, which is widely used in practice to construct a quantizer, but it cannot be assured that the obtained quantizer is optimum. We know only that the sequence of powers obtained by this algorithm is a non-increasing sequence! We have also seen that, when the number of quantization levels l is assumed to be high, we find explicitly, in contrast to the previous case, the expression for the optimum partition uniquely as a function of the probability density function $p_X(x)$.

In this section, a new scalar quantization mode, called scalar quantization with entropy constraint, is introduced. The minimization described above, through which the two necessary conditions for optimization are found, is concerned with a constrained minimization. The constraint was that the number of quantization levels L was fixed at 2^b. We can also seek to minimize σ_Q^2 relative to t^k and $\hat{x}^k$ with the constraint that the entropy $H(\hat{X})$ at the quantizer output must be less than or equal to b. As:

$$H(\hat{X}) \leq \log_2 L = b$$

we can hope that we will obtain a better-performing quantizer.

Let us begin with the case of a high number L of quantization levels: the probability density function can be assumed to be constant in the interval $[t^{k-1}, t^k]$ and the reproduction value $\hat{x}^k$ is in the middle of this interval.

4.4.2. *Lloyd–Max quantizer*

Recall equation [1.3] for the quantization error power:

$$\sigma_Q^2 = \frac{1}{12}\sum_{k=1}^{L} p_X(\hat{x}^k)\Delta^3(k)$$

as a function of $\Delta(k) = t^k - t^{k-1}$, with the interval length $[t^{k-1}, t^k]$.

Let us give a new explanation of equation [1.5] by recalling that a non-uniform scalar quantizer is a non-linear transform followed by an inverse transform. With the quantization step δ of the uniform quantizer, $f(x)$ the characteristic and $g(x)$ the derivative of $f(x)$, we have the relation:

$$\frac{\delta}{\Delta(k)} \approx f^{'}(\hat{x}^k) = g(\hat{x}^k) \qquad [4.7]$$

The quantization error power is written:

$$\sigma_Q^2 = \frac{1}{12}\sum_{k=1}^{L} p_X(\hat{x}^k)\frac{\delta^2}{g^2(\hat{x}^k)}\Delta(k)$$

Taking it to its limit, we find:

$$\sigma_Q^2 = \frac{\delta^2}{12}\int_{-\infty}^{+\infty}\frac{p_X(x)}{g^2(x)}\mathrm{d}x$$

Assume, for the purposes of simplification, that the support of $p_X(x)$ is the interval $[-A, +A]$ and that the probability density function is an even function. The quantization step δ of the uniform quantizer is given by:

$$\delta = \frac{2f(A)}{L}$$

The function $g(x)$ is not ordinary. It must satisfy the constraint:

$$\int_{-A}^{+A} g(x)\mathrm{d}x = \int_{-A}^{A} f^{'}(x)\mathrm{d}x = 2f(A)$$

The problem consists therefore in finding:

$$\bar{g}(x) = \frac{g(x)}{2f(A)}$$

to minimize:

$$\sigma_Q^2 = \frac{1}{12}\left(\int_{-A}^{+A}\frac{p_X(x)}{\bar{g}^2(x)}\mathrm{d}x\right)2^{-2b}$$

under the constraint:

$$\int_{-A}^{+A} \bar{g}(x)\mathrm{d}x = 1$$

Let us use the Hölder inequality which says that:

$$\int_{-A}^{+A} u(x)v(x)\mathrm{d}x \leq \left(\int_{-A}^{+A} u^a(x)\mathrm{d}x\right)^{1/a} \left(\int_{-A}^{+A} v^b(x)\mathrm{d}x\right)^{1/b}$$

provided that a and b are positive integers which satisfy:

$$\frac{1}{a} + \frac{1}{b} = 1$$

The inequality is replaced by equality when $u^a(x)$ is proportional to $v^b(x)$. Let us select:

$$u(x) = \left(\frac{p_X(x)}{\bar{g}^2(x)}\right)^{1/3}$$

$$v(x) = \bar{g}^{2/3}(x)$$

and $a = 3$ and $b = 3/2$. We find:

$$\int_{-A}^{+A} p_X^{1/3}(x)\mathrm{d}x \leq \left(\int_{-A}^{+A} \frac{p_X(x)}{\bar{g}^2(x)}\mathrm{d}x\right)^{1/3} \left(\int_{-A}^{+A} \bar{g}(x)\mathrm{d}x\right)^{2/3}$$

We therefore have:

$$\sigma_Q^2 \geq \frac{1}{12}\left(\int_{-A}^{+A} p_X^{1/3}(x)\mathrm{d}x\right)^3 2^{-2b}$$

with equality when:

$$\bar{g}(x) = \frac{p_X^{1/3}(x)}{\int_{-A}^{A} p_X^{1/3}(x)\mathrm{d}x}$$

We have arrived again at equation [1.4] which gives the minimum value of the quantization error power.

4.4.3. *Quantizer with entropy constraint*

We need to determine δ and $g(x)$ to minimize:

$$\sigma_Q^2 = \frac{1}{12}\int_{-\infty}^{+\infty} \left[\frac{\delta}{g(x)}\right]^2 p_X(x)\mathrm{d}x \qquad [4.8]$$

with a constraint set on the entropy of the quantizer output:

$$H(\hat{X}) \leq b$$

4.4.3.1. *Expression for the entropy*

The entropy of the quantizer output is expressed:

$$H(\hat{X}) = -\sum_{k=1}^{L} \text{Prob}(k) \log_2 \text{Prob}(k)$$

with:

$$\text{Prob}(k) = \text{Prob}\{X \in [t^{k-1}, t^k]\} = p(\hat{x}^k)\Delta(k)$$

We therefore have:

$$H(\hat{X}) = -\sum_{k=1}^{L} p_X(\hat{x}^k)\Delta(k) \log_2 p_X(\hat{x}^k)\Delta(k)$$

$$H(\hat{X}) = -\sum_{k=1}^{L} p_X(\hat{x}^k)\Delta(k) \log_2 p_X(\hat{x}^k) - \sum_{k=1}^{L} p_X(\hat{x}^k)\Delta(k) \log_2 \Delta(k)$$

or by exploiting equation [4.7]:

$$H(\hat{X}) = -\sum_{k=1}^{L} p_X(\hat{x}^k)\Delta(k) \log_2 p_X(\hat{x}^k) - \sum_{k=1}^{L} p_X(\hat{x}^k)\Delta(k) \log_2 \frac{\delta}{g(\hat{x}^k)}$$

Taking limits, we find:

$$H(\hat{X}) = -\int_{-\infty}^{+\infty} p_X(x) \log_2 p_X(x) \mathrm{d}x - \int_{-\infty}^{+\infty} p_X(x) \log_2 \frac{\delta}{g(x)} \mathrm{d}x$$

The first term depends only on the source. This is called the differential entropy and is conventionally written:

$$h(X) = -\int_{-\infty}^{+\infty} p_X(x) \log_2 p_X(x) \mathrm{d}x$$

The differential entropy characterized the quantity of information that a continuous memoryless source has. It shares most of the properties of the entropy for a discrete memoryless source except that it is not necessarily positive. For example, for a source which has a uniform distribution in the interval $[-A/2, A/2]$, we find:

$$h(X) = \int_{-\infty}^{+\infty} \frac{1}{A} \log_2 A \, \mathrm{d}x = \log_2 A$$

The sign of $h(X)$ depends on the support of $p_X(x)$. If A is less than 1 or greater than 1, the differential entropy will be negative or positive respectively.

4.4.3.2. *Jensen inequality*

This inequality is often used in information theory. It says that if X is a random variable in the interval $[a, b]$ and if $f(x)$ is a convex function in the same interval:

$$\mathbb{E}\{f(X)\} \geq f(\mathbb{E}\{X\})$$

Furthermore, if $f(x)$ is strictly convex, then the equality is true if and only if $X = \mathbb{E}\{X\}$ with a probability of 1, that is, if X is constant.

We now show the first result when X is a discrete random variable which takes the values x^1 and x^2 with probabilities $p_X(1)$ and $p_X(2)$. Since a convex function satisfies:

$$\lambda f(x^1) + (1-\lambda) f(x^2) \geq f(\lambda x^1 + (1-\lambda) x^2)$$

$\forall x^1, x^2 \in [a, b]$ and $0 \leq \lambda \leq 1$, we can write:

$$p_X(1) f(x^1) + (1 - p_X(1)) f(x^2) \geq f[p_X(1) x^1 + (1 - p_X(1)) x^2]$$

that is:

$$\mathbb{E}\{f(X)\} \geq f(\mathbb{E}\{X\})$$

For a random variable X which takes values $\{x^1 \cdots x^L\}$, we can assume that this property is true to order $k-1$ and write:

$$\sum_{i=1}^{k} p_X(i) f(x^i) = p_X(k) f(x^k) + (1 - p_X(k)) \sum_{i=1}^{k-1} \frac{p_X(i)}{1 - p_X(k)} f(x^i)$$

$$\sum_{i=1}^{k} p_X(i) f(x^i) \geq p_X(k) f(x^k) + (1 - p_X(k)) f\left(\sum_{i=1}^{k-1} \frac{p_X(i)}{1 - p_X(k)} x^i \right)$$

$$\sum_{i=1}^{k} p_X(i) f(x^i) \geq f(p_X(k) x^k + (1 - p_X(k))) f\left(\sum_{i=1}^{k-1} \frac{p_X(i)}{1 - p_X(k)} x^i \right)$$

$$\sum_{i=1}^{k} p_X(i) f(x^i) \geq f\left(\sum_{i=1}^{k} p_X(i) x^i \right)$$

Taking limits, this result generalizes to the case of a random variable with continuous values.

4.4.3.3. *Optimum quantizer*

Equation [4.8] can be written:

$$\sigma_Q^2 = \frac{1}{12}\mathbb{E}\left\{\left[\frac{\delta}{g(X)}\right]^2\right\}$$

Since the function x^2 is convex, we have:

$$\sigma_Q^2 \geq \frac{1}{12}\left(\mathbb{E}\left\{\frac{\delta}{g(X)}\right\}\right)^2$$

which is equal when $g(x)$ is a constant value written g_0 over the whole support of $p_X(x)$. This means that the best quantizer for the continuous source $X(n)$ is simply the uniform quantizer followed by entropy coding. The ratio δ/g_0 must satisfy the constraint:

$$h(X) - \log_2 \frac{\delta}{g_0} \leq b$$

which leads to:

$$\frac{\delta}{g_0} \geq \ 2^{h(X)-b}$$

The quantization error power is therefore minored by:

$$\sigma_Q^2 = \frac{1}{12}2^{2h(X)}2^{-2b} \tag{4.9}$$

4.4.3.4. *Gaussian source*

Let us show the expression for the differential entropy of a Gaussian source. We have:

$$h(X) = \int_{-\infty}^{+\infty} p_X(x)\left[\log_2\sqrt{2\pi\sigma_X^2} + \frac{x^2}{2\sigma_X^2}\log_2 e\right]\mathrm{d}x$$

$$h(X) = \log_2\sqrt{2\pi\sigma_X^2}\int_{-\infty}^{+\infty} p_X(x)\mathrm{d}x + \frac{\log_2 e}{2\sigma_X^2}\int_{-\infty}^{+\infty} p_X(x)x^2\mathrm{d}x$$

$$h(X) = \log_2\sqrt{2\pi\sigma_X^2} + \frac{\log_2 e}{2}$$

$$h(X) = \frac{1}{2}\log_2 2\pi e\sigma_X^2$$

Substituting into [4.9], we find:

$$\sigma_Q^2 = \frac{\pi e}{6}\sigma_X^2 2^{-2b}$$

The constant $\pi e/6$ is equal to 1.42. The gain carried by the quantizer with entropy constraint relative to the Lloyd–Max quantizer is:

$$\frac{\sqrt{3}\pi/2}{\pi e/6} = \frac{3\sqrt{3}}{e} = 1.91$$

which is 2.81 dB.

4.5. Capacity of a discrete memoryless channel

4.5.1. *Introduction*

If a channel, receiving an input symbol $X(n)$ from the alphabet $A_X = \{x^1 \cdots x^{L_X}\}$ at time n, produces an output symbol $Y(n)$ from the alphabet $A_Y = \{y^1 \cdots y^{L_Y}\}$, and if the output symbol depends only on the input symbol at the same time, we say that the channel is discrete and memoryless. This is shown as a diagram in Figure 4.5.

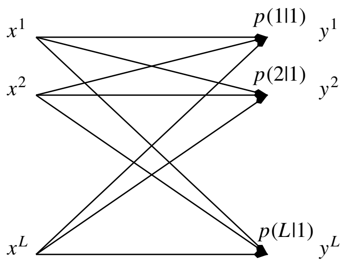

Figure 4.5. *Noisy channel*

The channel can be noisy. It is therefore natural to find out how to characterize the mapping of input symbols with output symbols by the set of conditional probabilities:

$$p_{Y|X}(j|i) = \text{Prob}\{Y(n) = y^j|X(n) = x^i\}$$

We assume that the two alphabets A_X and A_Y have the same number of symbols. The set of conditional probabilities can be put into the form of a square matrix. For a slightly noisy channel, this matrix is close to the identity matrix.

We then ask ourselves the following question: what is the information quantity which produces the event $\{Y(n) = y^j\}$ and concerns the event $\{X(n) = x^i\}$?

4.5.2. *Mutual information*

The mutual information of two events $\{X(n) = x^i\}$ and $\{Y(n) = y^j\}$ is the quantity:

$$I(x^i; y^j) = -\log_2 \frac{p_X(i)p_Y(j)}{p_{XY}(i,j)}$$

$$I(x^i; y^j) = -\log_2 \frac{p_Y(j)}{p_{Y|X}(j|i)} = -\log_2 \frac{p_X(i)}{p_{X|Y}(i|j)}$$

Mutual information[4] is zero if event $\{Y(n) = y^j\}$ is independent of event $\{X(n) = x^i\}$. It is at a maximum if $p_{Y|X}(j|i) = 1$, that is, if the channel has no errors for this pair. In this case, the mutual information is equal to the information itself $I(x^i)$. It therefore represents the amount of information produced by the received symbol y^j on the emitted symbol X^i.

The mutual information of two random variables X and Y is the expectation of $I(x^i; y^j)$. It is written:

$$I(X;Y) = \sum_{i=1}^{L_X} \sum_{j=1}^{L_Y} p_{XY}(i,j) I(x^i; y^j)$$

$$I(X;Y) = -\sum_{i=1}^{L_X} \sum_{j=1}^{L_Y} p_{XY}(i,j) \log_2 \frac{p_X(i)p_Y(j)}{p_{XY}(i,j)} \qquad [4.10]$$

Let us state a few properties of $I(X;Y)$. Note above all else that the mutual information is the relative entropy defined by equation [4.3] between the joint probability and the product of the marginal probabilities.

$$I(X;Y) = D(p_{XY} || p_X p_Y)$$

From that we deduce that $I(X;Y)$ is non-negative and is 0 if and only if the two random variables X and Y are independent. The mutual information $I(X;Y)$ measures the dissimilarity between the joint probability and what it would be if the two random variables were independent. It measures the amount of information that the two random variables carry together.

4. We must carefully note the different punctuation signs in the notation. Commas are conventionally associated with joint probabilities and semi-colons are used to indicate mutual information.

As equation [4.10] is written:

$$I(X;Y) = -\sum_{i=1}^{L_X}\sum_{j=1}^{L_Y} p_{XY}(i,j)\log_2 \frac{p_X(i)}{p_{X|Y}(i|j)}$$

$$I(X;Y) = -\sum_{i=1}^{L_X} p_X(i)\log_2 p_X(i) + \sum_{i=1}^{L_X}\sum_{j=1}^{L_Y} p_{XY}(i,j)\log_2 p_{X|Y}(i|j)$$

we find:

$$I(X;Y) = H(X) - H(X|Y)$$

The mutual information therefore measures the reduction in uncertainty of X given the knowledge of Y. The conditional entropy $H(X|Y)$ can be considered to be the mean uncertainty of the symbol emitted by the source after the symbol produced at the receiver has been specified. For a slightly noisy channel, the conditional entropy is almost zero. The mutual information is therefore maximum. It is practically equal to the source entropy. The mutual information characterizes the information transfer.

By developing equation [4.10], we can directly obtain:

$$I(X;Y) = H(X) + H(Y) - H(X,Y)$$

When $Y = X$, we have:

$$I(X;X) = H(X)$$

The expression for the mutual information, showing the conditional probabilities $p_{Y|X}(j|i)$ which characterize the channel and the source symbol probabilities, $p_X(i)$, is given by:

$$I(X;Y) = -\sum_{i=1}^{L_X}\sum_{j=1}^{L_Y} p_X(i)p_{Y|X}(j|i)\log_2 \frac{p_Y(j)}{p_{Y|X}(j|i)}$$

As:

$$p_Y(j) = \sum_{k=1}^{L_X} p_X(k)p_{Y|X}(j|k)$$

we find:

$$I(X;Y) = -\sum_{i=1}^{L_X}\sum_{j=1}^{L_Y} p_X(i)p_{Y|X}(j|i)\log_2 \frac{\sum_{k=1}^{L_X} p_X(k)p_{Y|X}(j|k)}{p_{Y|X}(j|i)}$$

4.5.3. *Noisy-channel coding theorem*

The channel capacity is the maximum mutual information for all possible source distributions:

$$C = \max_{p_X(i)} I(X;Y)$$

The capacity of a channel is the largest amount of information that can be transmitted through it. Since $I(X;Y) \geq 0$, $I(X;Y) = H(X) - H(X|Y)$, and $H(X|Y) \geq 0$, we deduce that:

$$C \leq \log_2 L$$

The source is characterized by its entropy rate $\bar{H}(X)$. The channel is characterized by its capacity C. We can show that if:

$$\bar{H}(X) \leq C$$

a transmission with as low an error probability as we want is possible. We can also show that if:

$$\bar{H}(X) > C$$

a transmission with as low an error probability as we want becomes impossible.

4.5.4. *Example: symmetrical binary channel*

A diagram of the channel is shown in Figure 4.6. It is characterized by the matrix:

$$\begin{bmatrix} p_{Y|X}(1|1) & p_{Y|X}(2|1) \\ p_{Y|X}(1|2) & p_{Y|X}(2|2) \end{bmatrix} = \begin{bmatrix} 1-p_e & p_e \\ p_e & 1-p_e \end{bmatrix}$$

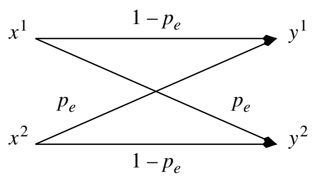

Figure 4.6. *Symmetrical binary channel*

To calculate the channel capacity, we maximize the mutual information as a function of the probability $p = p_X(1)$. We use the equation:

$$I(X;Y) = H(Y) - H(Y|X)$$

As:

$$H(Y|X) = pH(Y|x^1) + (1-p)H(Y|x^2)$$

and:

$$H(Y|x^1) = H(Y|x^2) = -(1-p_e)\log_2(1-p_e) - p_e \log_2 p_e$$

$$H(Y|x^1) = H(Y|x^2) = f(p_e)$$

we notice that:

$$H(Y|X) = pf(p_e) + (1-p)f(p_e) = f(p_e)$$

is independent of p. On the other hand, as:

$$H(Y) = f[p_Y(1)] = f[p(1-p_e) + (1-p)p_e]$$

and the function $f(x)$ reaches its maximum at $x = 1/2$ which has $p = 1/2$, we find $H(Y) = 1$. The capacity of the symmetrical binary channel is therefore:

$$C = 1 - f(p_e)$$

When the channel is noiseless, the capacity is 1. When $p_e = 1/2$, the capacity is zero. In general, the channel capacity calculation is much more difficult than in this case.

4.6. Coding a discrete source with a fidelity criterion

4.6.1. *Problem*

The previous theorem indicates that if the channel capacity is too low when taking account of the entropy rate of the source, transmission is not possible with an error probability as low as desired but, in fact, it is the practical case which is of interest. At the cost of a certain amount of distortion, we wish to compress a signal. Is it possible, in this case, to evaluate the system performance? A function known as the rate–distortion function allows us to find the smallest tolerable distortion when $\bar{H}(X) > C$. This theory shows that this function exists but does not produce a method by which a code with these performance characteristics can be constructed.

Instead of optimizing across all the deterministic codes which have a given structure while respecting a constraint of the transmission bit rate, we are interested, as is the case throughout this chapter, in codes characterized by random functions and a constraint set on the information bit rate.

4.6.2. *Rate–distortion function*

Consider a discrete source characterized by the alphabet $A_X = \{x^1 \cdots x^{L_X}\}$. Recall that inequalities exist between the source entropy $H(X)$, if its possible memory is not taken into account, the source entropy rate $\bar{H}(X)$, and the number of elements L_X in the alphabet A_X:

$$0 \leq \bar{H}(X) \leq H(X) \leq \log_2 L_X$$

Only the most simple case is examined here to find the coding for a memoryless discrete source. Let us introduce a new alphabet $A_{\hat{X}} = \{\hat{x}^1 \cdots \hat{x}^{L_{\hat{X}}}\}$ which is called the reproduction alphabet. The coding operation can be represented by the same diagram shown in Figure 4.5 if we replace y^k by $\hat{x}^k$. Each branch is characterized by the transition probabilities:

$$p_{\hat{X}|X}(j|i) = \text{Prob}\{\hat{X}(n) = \hat{x}^j | X(n) = x^i\}$$

Furthermore, each branch must have associated with it a measure of distortion, which is assumed for simplicity to have no effects from memory (or anticipation). This is known as the distortion measure per letter and is noted as $d(x^i, \hat{x}^j)$. Recall the expression for the mutual information:

$$I(X;\hat{X}) = -\sum_{i=1}^{L_X}\sum_{j=1}^{L_{\hat{X}}} p_X(i)p_{\hat{X}|X}(j|i)\log_2 \frac{\sum_{k=1}^{L_X} p_X(k)p_{\hat{X}|X}(j|k)}{p_{\hat{X}|X}(j|i)}$$

We need to find the optimum reproduction alphabet. This means that the degrees of freedom are, in this problem, the conditional probabilities. The probabilities $p_X(i)$ are set by the source. To show this, the mutual information is denoted:

$$I(p_{\hat{X}|X}) = I(X;\hat{X})$$

The mean distortion is written:

$$\bar{d}(p_{\hat{X}|X}) = \sum_{i=1}^{L_X}\sum_{j=1}^{L_{\hat{X}}} p_{X\hat{X}}(i,j)d(x^i, \hat{x}^j)$$

$$\bar{d}(p_{\hat{X}|X}) = \sum_{i=1}^{L_X}\sum_{j=1}^{L_{\hat{X}}} p_X(i)p_{\hat{X}|X}(j|i)d(x^i, \hat{x}^j)$$

The function known as the rate–distortion function is:

$$b(D) = \min_{p_{\hat{X}|X} \in Q} I(p_{\hat{X}|X})$$

where Q is the set of all the conditional probabilities which satisfy:

$$\bar{d}(p_{\hat{X}|X}) \leq D$$

The rate–distortion function depends only on the source.

To generalize these results for sources with memory, we need to group N input symbols into vector form:

$$X_N = [x(mN) \cdots x(mN + N - 1)]^t$$

and to define a set of functions $b_N(D)$. We can show that:

$$b_N(D) \geq b_{N+1}(D)$$

If zero distortion is required, the coding must be lossless. We have $b(0) = H(X)$. If D increases, the set Q increases as well which means that the rate–distortion function is non-increasing. More precisely, we can show that it is a continuous, convex and strictly decreasing function. If a distortion greater than or equal to the power of the source is acceptable (in the case of a quadratic distortion measure), there is not even a need to encode it. We have $b(\sigma_X^2) = 0$.

4.6.3. *Theorems*

4.6.3.1. *Source coding theorem*

If we know the rate–distortion function $b(D)$, we can show that for any $D \geq 0$ and for any $\epsilon > 0$, a vector quantizer exists which comprises L N-dimensional vectors which satisfy:

$$\frac{1}{N} \log_2 L < B(D) + \epsilon$$

and give rise to a mean distortion less than or equal to $D + \epsilon$.

4.6.3.2. *Combined source-channel coding*

The noisy-channel coding theorem says that if the channel capacity is high enough, that is, greater than or equal to the entropy rate of the source, there is no problem. If this capacity is too low, the new result shows that we can realize a compression while accepting a distortion such that $b(D) = C$, then carrying out a channel coding with as low an error probability as required.

4.6.4. *Special case: quadratic distortion measure*

4.6.4.1. *Shannon's lower bound for a memoryless source*

For a quadratic distortion measure, we can show that, for a non-Gaussian memoryless source of power σ_X^2, the rate–distortion function is mirrored by:

$$b(D) \geq h(X) - \frac{1}{2} \log_2 2\pi e D$$

The following inequality gives the lower bound of the quantization error power for a memoryless Gaussian source:

$$b = \frac{1}{2}\log_2 2\pi e\sigma_X^2 - \frac{1}{2}\log_2 2\pi e\sigma_Q^2 = \frac{1}{2}\log_2 \frac{\sigma_X^2}{\sigma_Q^2}$$

so that:

$$\sigma_Q^2 = \sigma_X^2\, 2^{-2b} \qquad [4.11]$$

The scalar quantizer with entropy constraint has a quantization error power of $\pi e/6 = 1.42$ times this limit. We can say that the scalar quantizer with entropy constraint is at:

$$\frac{1}{2}\log_2 \frac{\pi e}{6} = 0.25 \text{ bit}$$

of the theoretical bound. The curve $b(D) + 0.25$ is sometimes referred to as the Gish-Pierce asymptote.

This result is interesting since it shows that all memoryless sources with power σ_X^2 have a rate–distortion function below that of Gaussian sources with the same power when the distortion measure is the quadratic error. The Gaussian source is therefore the most difficult source to reproduce.

4.6.4.2. *Source with memory*

In the same way, for a source with memory we can generalize the preceding result without too much difficulty. The differential entropy rate of a continuous source, as it is known, is the limit:

$$\bar{h}(X) = \lim_{N\to\infty} \frac{1}{N} h(X_N)$$

where:

$$h(X_N) = -\int_{R^N} p_{X_N}(x)\log_2 p_{X_N}(x)\mathrm{d}x$$

and where $p_{X_N}(x)$ is the joint probability density function of the random vector which is composed of N consecutive symbols. For a zero-mean Gaussian source, with power σ_X^2 and normalized autocorrelation matrix Γ_X, the joint probability density function is written:

$$p_{X_N}(x) = \frac{1}{(2\pi\sigma_X^2)^{N/2}\sqrt{\det \Gamma_X}} e^{-(x^t\Gamma_X^{-1}x)/2\sigma_X^2}$$

We therefore have:

$$\begin{aligned} h(X_N) &= \frac{1}{2}\int_{-\infty}^{+\infty} p_X(x)\log_2(2\pi\sigma_X^2)^N \det\Gamma_X \mathrm{d}x \\ &\quad+\frac{\log_2 e}{2\sigma_X^2}\int_{-\infty}^{+\infty} p_X(x)x^t\Gamma_X^{-1}x\mathrm{d}x \\ h(X_N) &= \frac{1}{2}\log_2(2\pi\sigma_X^2)^N\det\Gamma_X + \frac{\log_2 e}{2\sigma_X^2}\mathbb{E}\{x^t\Gamma_X^{-1}x\} \end{aligned} \qquad [4.12]$$

Since we can demonstrate that:

$$\mathbb{E}\{x^t\Gamma_X^{-1}x\} = N\sigma_X^2$$

we find:

$$h(X_N) = \frac{1}{2}\log_2(2\pi e\sigma_X^2)^N\det\Gamma_X$$

$$\bar{h}(X) = \lim_{N\to\infty}\frac{1}{2}\log_2 2\pi e\sigma_X^2(\det\Gamma_X)^{1/N}$$

$$\bar{h}(X) = \frac{1}{2}\log_2\frac{2\pi e\sigma_X^2}{G_p(\infty)}$$

as a result of relation [1.12]. From this we can deduce that the lower bound of the quantization error power for a Gaussian source with memory is:

$$\sigma_Q^2 = \frac{\sigma_X^2}{G_p(\infty)}\,2^{-2b} \qquad [4.13]$$

4.6.5. *Generalization*

Consider equation [4.11]. From it we can directly deduce the minimum number of bits necessary to quantize a source with a signal-to-noise ratio given by:

$$b \geq \frac{1}{2}\log_2\frac{\sigma_X^2}{\sigma_Q^2} \qquad [4.14]$$

Equation [4.14] allows a first generalization as we have just seen. We obtain equation [4.13]. In practice, we have seen that several methods exist which allow us to make use of correlation (whitening filter, vector quantization, transformation or filter banks, etc.). The asymptotic value $G_p(\infty)$ of the prediction gain is uniquely a function of the spectral density $S_X(f)$ of the source:

$$G_p(\infty) = \frac{\int_{-1/2}^{1/2} S_X(f)\mathrm{d}f}{e^{\int_{-1/2}^{1/2} \log_e S_X(f)\mathrm{d}f}} \qquad [4.15]$$

Equations [4.13] and [4.15] enable us to obtain the minimum number of bits necessary to quantize a correlated source:

$$b \geq \frac{1}{2}\int_{-1/2}^{+1/2} \log_2 \frac{S_X(f)}{\sigma_Q^2}\mathrm{d}f \qquad [4.16]$$

A second generalization is necessary since, in the framework of coding audio-frequency signals, we will see that minimizing the power of the quantization noise is not the most important problem. We will look for the required bit rate which is sufficient for the noise power spectral density to be lower than the power spectral density limit produced by a psychoacoustic model. We can show [BER 71] that equation [4.16] generalizes in the following form:

$$b \geq \frac{1}{2}\int_{-1/2}^{+1/2} \max\left[0, \log_2 \frac{S_X(f)}{S_Q(f)}\right] \mathrm{d}f \qquad [4.17]$$

Intuitively, this equation is deduced directly from [4.16] or [4.14] by observing that the need to quantize the signal in frequency bands where $S_X(f) \leq S_Q(f)$ is useless, and also in "elementary" frequency bands $[f, f+df[$ which satisfy $S_X(f) > S_Q(f)$, the necessary "bit density" is given by:

$$\frac{\mathrm{d}b}{\mathrm{d}f} \geq \frac{1}{2}\log_2 c(1)\ \frac{S_X(f)}{S_Q(f)}$$

The theory therefore indicates that to encode a signal, when the power spectral density of the signal and tolerable noise level are known, we must determine frequency bands $[f_k, f_k^{'}]$ such that $S_X(f) \geq S_Q(f)$, and then allocate the available binary resources as a function of the ratio $S_X(f)/S_Q(f)$.

PART 2
Audio Signal Applications

Chapter 5

Introduction to Audio Signals

5.1. Speech signal characteristics

Speech signals are very complex with numerous characteristics. Here, we content ourselves with presenting a few properties that are significant in signal compression.

Figure 5.1 shows a speech signal, 0.5 s in length, from a female speaker in the time domain (Figure 5.1(a)) and in the frequency domain (Figure 5.1(b)).

The left-hand graph shows that the signal is obviously not stationary, but it can be considered to be locally stationary for periods of the order of a few dozen milliseconds. In speech coding, analysis frames of 20 ms are the standard choice.

Different sound types can be distinguished: voiced sounds, unvoiced sounds[1], and plosive sounds. Compression of voiced and unvoiced sounds can be carried out in good conditions as we will see below. However, there is no equivalent for plosive sounds and for transitions between phonemes.

The third characteristic, which is very important as we will see, is the existence of a simple and an effective production model. Let us examine the graph in Figure 5.1(b), which gives two spectral estimates. The first estimate is realized by calculating a periodogram, that is, by taking the square of the modulus of the discrete Fourier transform with N samples. This calculation yields a good approximation of the true, but inaccessible, power spectral density. This spectral density is known as $\hat{S}_1(f)$. It is represented by a curve with numerous peaks. The second estimate $\hat{S}_2(f)$ uses an

1. The samples shown here do not include this type of sound.

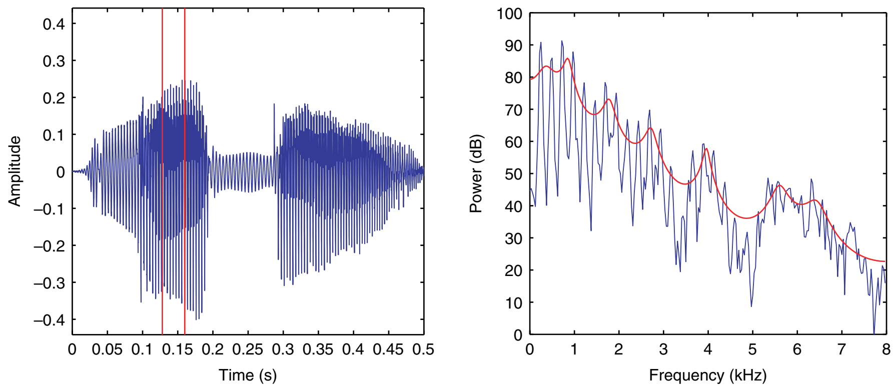

Figure 5.1. *Example of a speech signal (a) in the time domain and (b) in the frequency domain. The samples from which the two spectral estimations were taken are between the two vertical lines on the left-hand graph*

autoregressive model, that is, assumes that the analyzed signal is the result of filtering white noise through an autoregressive filter. The curve $\hat{S}_2(f)$ is much smoother. It corresponds to the spectral envelope. A closer inspection of this curve shows that $\hat{S}_1(f)$ can be seen, to the first approximation, as $\hat{S}_1(f)$ is multiplied by a set of regularly spaced impulses of the same power[2]. This means that it seems reasonable to model a voiced sound by the output of an autoregressive filter which has a Dirac comb as its input. For unvoiced sounds, listening tests show that the same type of filter can be used reasonably well provided that the Dirac comb is replaced by white noise.

5.2. Characteristics of music signals

The graph in Figure 5.2(a) shows the signal produced by a violin during a period of half a second. As for speech signals, we notice significant variations in the instantaneous power. These can be considerable, up to 90 dB, and not confined to particularly percussive signals. This variation can happen, for example, in a Mahler symphony between pianissimo and fortissimo passages. The illustration of the spectrum on the right-hand graph, between 0 and 20 kHz (this signal was sampled at 44.1 kHz), conveys the instrument's timbre. The dominant spectral components, the *partials*, may or may not be harmonically linked, as in the case shown for the violin signal. It must be noted that a simple production model no longer exists for this task. Musical sounds are produced in ways that are too different.

2. All at least up to approximately 2 kHz. The gap between impulses is called the fundamental frequency, the *pitch* frequency.

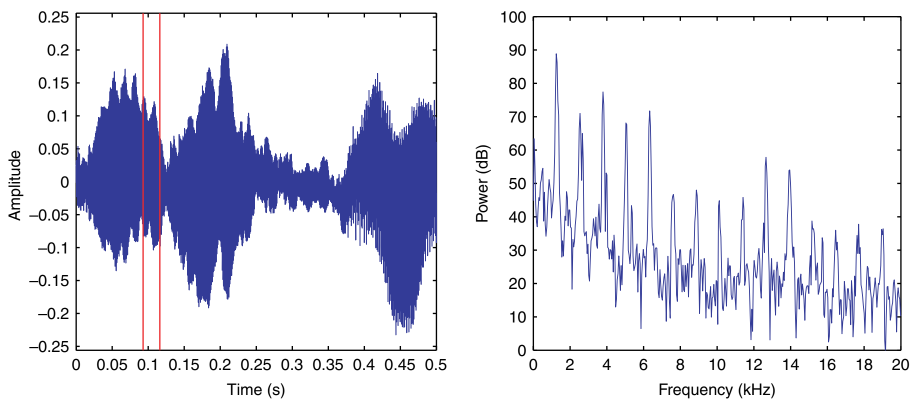

Figure 5.2. *Example of a violin sound signal (a) in the time domain and (b) in the frequency domain*

5.3. Standards and recommendations

5.3.1. *Telephone-band speech signals*

The different standards and recommendations are classified according to their applications.

5.3.1.1. *Public telephone network*

Several "recommendations" have been defined for the public telephone network during the 30 years in the framework of the International Telecommunication Union, Telecommunication standardization sector (ITU-T).

Since 1972, the G.711 recommendation has specified a coding by *pulse-code modulation* (PCM), which corresponds to a bit rate of 64 kb/s: the sample amplitudes are quantized on 8 bits after non-linear compression. The majority of switched telephone networks in the world use this coding.

Since 1984, the G.721 recommendation has defined an adaptive, differential PCM (ADPCM) coding, which corresponds to a bit rate of 32 kb/s: the sample amplitudes are no longer directly quantized; instead the difference between the amplitude and the predicted value determined by an adaptive-type filter is quantized. This coding is often used for transmission via submarine cables or satellites. It is also used in the European standard for cordless telephones, *Digital Enhanced Cordless Telecommunications* (DECT). For the bit rates of 16/24/32/40 kb/s, there is a multi-bit-rate version (G.726) with the preceding coding.

A 16-kb/s coder based on modeling techniques and vector quantization was selected by ITU-T in 1991. This recommendation (G.728) is also called *low-delay code-excited linear predictive* (LD-CELP) coder, which shows that it is a CELP coder (presented at length below) and that it has a low reconstruction delay, which is a particularly an important property for a telephone conversation.

In 1995, an international competition selected a 8-kb/s coder called G.729 coder, which combined the *Algebraic Code Excited Linear Predictive* (ACELP) coder, developed at the University of Sherbrooke (Canada) and in the France Telecom laboratories (Lannion), and the Nippon Telegraph & Telephone Corp. (NTT) coder.

Finally, a 6.3-kb/s coder, G.723.1, specifies videophone sound.

5.3.1.2. *Mobile communication*

Communication via mobile devices has developed a great deal over the past 20 years. The nature of the transmission channel, a radio link, demands that the transmitted signal's bandwidth is as economic as possible to allow a large number of users on the network.

Since 1989, the European standard called Group Special Mobile (GSM) 06.10 has been in place, selected by the *European Telecommunications Standard Institute* (ETSI). A standard that covers North America, called IS 54, was defined in 1990 by the *Telecommunication Industry Association* (TIA). The Itineris service was opened in July 1992 by France Telecom. This first generation was founded on time division multiple access (TDMA) techniques and source coders[3], *regular pulse excitation–long-term prediction* (RPE-LTP) in Europe at 13 kb/s, *vector sum excited linear predictive* (VSELP) coder in North America at 8 kb/s.

A new standard (GSM 06.60) was defined in 1996 to improve the quality of the GSM coder. This is the *Enhanced Full-Rate* (EFR) coder [EUR 96]. In the absence of any errors, its quality can be said to be equivalent to that of G.726 at 32 kb/s.

Since 1999, a new coder (GSM 06.90) has existed, which realizes the bit rate split between source coding and channel coding depending on the propagation conditions. This uses a variable bit rate *Adaptive Multi-Rate* (AMR) coder between 4.75 and 12.2 kb/s (in reality between 11.4 and 22.8 kb/s).

New standardizations *3rd Generation Partnership Project* (3GPP) appeared in light of the emergence of mobile communication systems said to be of the third-generation *Universal Mobile Telephone System* (UMTS). Collaboration between

3. All the specified bit rates in this paragraph concern source coding. More or less the same bit rate must be added for channel coding, for example, 9.8 kb/s for GSM, the total bit rate is therefore 22.8 kb/s.

ITU and ETSI has enabled the definition of the G.722.2 coder, which is a wideband speech coder.

5.3.1.3. *Other applications*

In other applications, for example, for secure communications between intergovernmental organizations, military applications (NATO standard), and mobile communications via satellite, a low bit rate is demanded at the cost of a significant signal degradation. The US Department of Defense (US DoD) standardized a 4.8-kb/s coder, called FS-1016, in 1991 to replace the former federal standard FS-1015 or LP-C10 defined in 1976 with a very mediocre quality.

5.3.2. *Wideband speech signals*

The purpose of transmitting wideband speech signals (f_e = 16 kHz) is to obtain a cleaner and more intelligible reconstructed speech signal. Applications are audiovisual conferences, videophone, and loudspeaker telephones.

A ITU-T standard at 64 kb/s, called G.722, was specified in 1986. This coder is based on a coding using two sub-bands each containing a ADPCM coder (G.721 standard). To be able to simultaneously transmit data in the same channel, a reduced bit rate of 56 kb/s is also possible.

A new coder (G.722.1) was defined in 1999 for 24- and 32-kb/s bit rates.

A new ITU-T standardization in 2002, the G.722.2 coder, was accepted beforehand by the 3GPP standardization organization under the name AMR-WB. A full description is available in [BES 02].

5.3.3. *High-fidelity audio signals*

The standard format for music signals is usually the compact disc: the signal is sampled at 44.1 kHz then scalar quantized on 16 bits. The corresponding bit rate is therefore 705 kb/s (in mono, or 1.4 Mb/s in stereo). Regarding the format, the disc itself is noticeably fragile, and it can be scratched, and consequently, the bits must be protected. Two stages of Reed–Solomon correction codes were introduced.

We can also note that other formats exist [KON 03]: Super Audio CD (SACD) and DVD-Audio. The first is the result of sampling the signal at 2.82 MHz on 1 bit (sigma/delta modulation), and the second uses a sampling frequency of 48, 96, or 192 kHz and quantization on 24 bits. These two formats, currently in competition with each other, allow data storage on 2–6 channels. They provide approximately the same quality (an improvement in dynamics of approximately 20 dB in power relative to that of a CD) and the same capacity (approximately 5 GB).

5.3.3.1. *MPEG-1*

The first application that required a compression operation was broadcasting audio signals in numeric form (DAB). These developments led to the audio part of the international standard ISO/CEI 11172 in 1992 [NOR 93], better known under the name MPEG-1 Audio. Of the two coders "in the final round," the MUSICAM and ASPEC coders, the former was selected. The coder consists of three *layers* with equivalent qualities at 192, 128, and 96 kb/s bit rates but with increased complexity. The well-known MP3 format is really MPEG-1 layer 3. Although it has the greatest complexity, this complexity is now considered to be very reasonable.

5.3.3.2. *MPEG-2*

In the audio domain, there is no great difference between the MPEG-1 coder and the MPEG-2 coder, standardized in 1994 (ISO/IEA 13818) for HDTV, DVD, and minidisc[4]. The second is a multi-track (5.1) extension of the first. It introduces a great flexibility in the choices of sampling frequencies and bit rates. This new coder is also *backward compatible*, that is, the MPEG-1 decoder can interpret the MPEG-2 stream which is less usual than forward compatibility, through which the MPEG-2 decoder can also decode the MPEG-1 stream. This property is important in a broadcasting network because it is easy to switch from one coder to another at the transmitter, while we cannot expect millions of television viewers to change their decoders abruptly. This backward compatibility penalizes the compression system performance so much that new research was launched without these constraints and, in 1997, resulted in a new standard [INT 97] called the *Audio Advanced Coder* (AAC), which ensures a "transparent" quality with higher compression rates. It is reputed to be transparent at a bit rate of 384 kb/s in the 5.1 configuration, which corresponds approximately to a transparency at 64 kb/s in mono. It is the highest performance coder at the time of writing and is the coder used in iPods.

Note that other "proprietary" coders exists, for example, Dolby AC3, which appeared in 1995 and Sony ATRC3 for minidisc (which has disappeared).

5.3.3.3. *MPEG-4*

MPEG-4 is a more and more ambitious standard. It aims to represent sounds (speech and music) of natural origin (from a microphone) or synthetic origin (produced by a machine). It also aims to define sound objects that are suitable to being manipulated in such a way as to create sound scenes. The first standardization was in

4. Audio professionals, at the beginning of the 1980s, seemed to be very reticent about introducing numeric signal processing for music, a reticence which has been progressively won over by the flexibility and performance of these processing techniques. These professionals, especially in television, came round to compression techniques toward the end of the 1980s.

1998 [INT 98]. This standardization has since been subject to several revisions. The following presentation is taken from the “Draft International Standard” dated 2005 [INT 05].

For sounds of natural origin, MPEG-4 does not produce coders that necessarily ensure transparency or even near transparency as the MPEG-1 and MPEG-2 standards do. It defines a family of coders from 2 kb/s to several hundred kilobytes per second to assure the best possible quality for a given bit rate. Below 64 kb/s and *all the more so* for very low bit rates, this is not just a dream: the quality can be reasonable, and it is said to be “intermediate” or even mediocre. MPEG-4 also defines a family of hierarchical coders (*scalable*) to be able to modify the bit rate in case of network congestion. The binary chain is constructed in several layers of such a type that the decoder can process all or part of the binary chain (to the detriment of quality).

MPEG-4 provides a toolbox of several compression algorithms.

From 2 to 24 kb/s for telephone-band or wideband speech signals, two coding techniques are used:

– The MPEG-4 *Harmonic Vector eXcitation Coding* (HVXC) operates at bit rates between 2 and 4 kb/s. It processes telephone-band speech signals. It accepts *time-stretching* manipulations (which modify the length of a recording without altering the pitch), which enables, for example, the rapid consultation of large speech databases.

– The MPEG-4 CELP coder takes over from 4 kb/s. It can process speech signals either in telephone band or in wideband. Telephone-band speech signals are processed by a standard CELP coder, which is very similar to the ITU-T G.729 coder. For wideband speech signals, an initial coder makes use of the production model as used in processing the frequency band [0–4] kHz. A second coding makes use of a hearing model as used in the MPEG-1 or MPEG-2 coder which takes care of the frequency band [4–7] kHz.

Using 6 kb/s to several hundred kilobytes per second for monophonic, stereophonic, or multi-channel (generally in 5.1 format) music, the MPEG-4 AAC is almost a replica of the MPEG-AAC. For bit rates higher than 64 kb/s per channel, the coder satisfies the transparency criterion. For lower bit rates, it finds the best quality for a given bit rate. There are several variations of this coder:

– the *Low Delay* version for interactive communications;

– a *Bit Slice Arithmetic Coding* (BSAC) version for a very fine “granularity” (1 kb/s);

– a *Transform Weighted INterleave-Vector Quantization* (TWIN-VQ version), ensures a better performance than the AAC between 6 and 16 kb/s.

To decrease the bit rate without affecting the quality, a new tool called *Spectral Band Replication* (SBR) tool, which is the result of several works, for example, has been recently added [COL 02]. In this method, high-frequency contributions to the music are reconstructed from the bass frequency spectrum with very little additional information. Hence, a bit rate of 24 kb/s can be obtained using 22 kb/s for the bass frequencies (the AAC codes a downsampled signal by a factor of 2) and 2 kb/s to reconstruct the high frequencies. The AAC's performance is clearly improved at low bit rates. This coder is a candidate for several standards: *Digital Radio Mondial*[5], 3GPP, and Digital Video Broadcasting (DVB).

In the case of music signals, parametric coders must be added to the above list:

– The MPEG-4 *Harmonic and Individual Line plus Noise* (HILN) coding codes music signals in the range of 4 to 6 kb/s bit rates. It uses a signal representation based on sinusoidal waveforms, on groups of harmonically linked sinusoidal waveforms, and on noise components. The individual parameters of each object are coded individually. This produces high flexibility in reconstruction. As for the speech coder HVXC, it can accept *time-stretching* manipulations.

– This idea is generalized in the case of the MPEG-4 *SinuSoidal Coding* (SSC), which takes advantage of objects to decompose the signal: sinusoids, transitories, noise, and also a spatial image in the case of stereo signals or even multi-channel as we will see in Chapter 8. It is capable of coding a full band signal at a reduced bit rate with a very high quality.

When the MPEG-4 AAC is used with SBR tools and *Parametric stereo*, it is called MPEG-4 AAC+ or HE-AACv2 (*High Efficiency*) (http://en.wikipedia.org/wiki/AacPlus).

Finally, we must mention the MPEG-4 *Scalable Lossless coding* (SLS) coder. This is a lossless and scalable bit rate coder with a fine granularity. We are no longer concerned with the previous bit rate range. The bit rates of interest here are from 64 to 705 kb/s for a monophonic signal. Note that lossless coding does not ensure particularly a high compression rate, around the order of 2. The lower bit rate limit is therefore of the order of 300 kb/s to ensure lossless coding.

For synthetic origin sounds, we find the following:

– A speech synthesis algorithm (*Text-to-Speech* synthesis) is a basic tool for multimedia applications. The transmitted speech information may be elements of the international phonetic alphabet or text written in any language.

– *Structured Audio Orchestra Language* (SAOL) is a language to create music. MPEG-4 allows the MIDI format to be used.

– MPEG-4 standardizes the way of describing a scene. It defines the area in a coordinates system where a sound is (or moves) (in-scene navigation). It also describes

5. DRM is not to be confused with *Digital Rights Management* at the heart of MPEG-21!

the way in which the appearance of each object changes. These changes may concern prosody for speech or reverberation and spatialization for music.

The above description, principally concerned with compression, remains silent on the vast potential of the MPEG-4 standard. To hint at this potential, we provide an example from [INT 05].

Assume that, in a particular application, we wish to make a good-quality transmission of the sound created by a person speaking in a reverberating environment with stereo music in the background. Using a traditional approach, an audio coder functioning at 32 kb/s per channel, for example, will suffice. With MPEG-4, the global signal can be represented as the composite of several objects: the sound produced by a person is passed through a reverberator, to which we add an extract of synthetically produced music. The voice is transmitted using the CELP tool at a bit rate of 16 kb/s, the music is synthesized using the Structured Audio (SA) tool at a bit rate of 2 kb/s, and 1 or 2 kb/s is added to describe the stereo and reverberation aspects. In conclusion, for equivalent qualities, the MPEG-4 approach costs less than 20 kb/s, whereas the more traditional approach requires 64 kb/s. Furthermore, the person listening at the receiver may only want to hear the person alone, which is only possible through the MPEG-4 approach.

5.3.3.4. *MPEG-7 and MPEG-21*

MPEG-7 does not handle compression problems. It is concerned with indexing. MPEG-21 centers its works on problems linked to the security of multi-media content. The techniques used are, for example, watermarking.

5.3.4. *Evaluating the quality*

The quality of reconstructed audio signals (speech or music) cannot be appreciated by using objective criteria such as the signal-to-noise ratio. As we will see more particularly when we look at music coding, at equivalent noise levels (from compression), some coders produce good-quality codings and others bad-quality codings. The spectral form of noise plays a very important role in its perception. We are therefore obliged to content ourselves with subjective tests (listening), but these subjective tests are said to be formal because the protocols are very tightly defined.

For intrinsically mediocre-quality speech coders, intelligibility tests are used: judging by absolute categories (*Absolute Category Rating*), degradation categories (*Degradation Category Rating*), and so on.

For audio coders, we generally want them to be good quality. Hence, transparency is required. To compare coders with one another, a method known as *double-blind with triple stimulus and dissimulated reference* is used. This is the ITU-R BS.1116 recommendation. Short recordings (between 5 and 10 s) of selected music pieces (from the MPEG database) are played to young (from 30 years old hearing degrades) musicians (well-practiced ears are required). Each recording is repeated three times. Two possibilities are offered: sequence A/B/A or A/A/B where A represents the original signal and B is the coded and reconstructed signal. The first required response is: is B in second or third position? An opinion is required (from 1 to 5) on B: is the compression noise imperceptible (5), perceptible but not annoying (4), slightly annoying (3), annoying (2), very annoying (1)? Statistical processing is carried out to enable an objective comparison between coders.

For audio coders with bit rates between 20 and 64 kb/s which oblige us to content ourselves with an "intermediate" (acceptable) quality, the *MUlti Stimulus test with Hidden Reference and Anchor* (MUSHRA) method is generally used. This is the recommendation ITU-R BS.1534-1.

Note that there are even objective tests that can give significant results. For example, we refer to the PEAK algorithm at the heart of the ITU-R BS.1387-1 recommendation [UNI 01].

Chapter 6

Speech Coding

6.1. PCM and ADPCM coders

For telephone-band speech signals, there is a significant reduction in the band relative to that of natural speech. As a result, significant loss in quality occurs. A 16-bit quantization that ensures (along with a sampling frequency of 44.1 kHz) "compact disc" quality is therefore unnecessary. We can say that a 12-bit quantization is "homogeneous" with a sampling frequency of 8 kHz. The reference bit rate for telephone-band speech signals is therefore of the order of 100 kb/s.

The 64 kb/s PCM ITU-T G.711 coder realizes a non-uniform scalar quantization like that which was presented (very briefly) in section 1.2.3. A straightforward scalar quantization is not well adapted to signals which present significant instantaneous variations in power. We can show that to maintain a roughly constant signal-to-noise ratio, the adapted non-linear transform is logarithmic in nature. This transformation is generally approximated by straight segments. This is known as "A-law" in Europe and "μ-law" in the USA.

The operating principle of the 32 kbit/s ADPCM ITU-T G.726 coder shown in Figure 1.4 corresponds to a closed-loop predictive scalar quantizer. There are several possible linear prediction models. The G.726 coder uses an ARMA(2.6) model in which the two specified coefficients of the AR part and the six coefficients of the MA part are updated at the sample rate. To estimate these coefficients, a gradient method is used, but in reality this means that a simplified gradient method is used to speed up the calculation. The operational details have no pedagogic value and are not presented here.

As the prediction error power is approximately proportional to the signal power, this coder introduces a scale factor $g(n)$ such that $2^{-g(n)}y(n)$ is of constant power. In reality, $g(n)$ is recursively estimated using a recurrence equation in the form $g(n) = (1-\alpha)\, g(n-1)+\alpha\, w(n)$, where the parameter $\alpha \approx 10^{-6}$ characterizes the adaptation speed of $g(n)$.

6.2. The 2.4 bit/s LPC-10 coder

This coder has little practical relevance. However, it is of great pedagogic interest because it is the basis of real speech coders.

Consider the diagram in Figure 6.1 where $x(n)$ is the original speech signal, $y(n)$ is the output signal from the *analysis filter*, $\hat{y}(n)$ is the input to the *synthesis filter*, and $\hat{x}(n)$ is the reconstructed speech signal. The LPC-10 coder calculates the filter coefficients:

$$A(z) = 1 + a_1 z^{-1} + \cdots a_P z^{-P}$$

from the original signal and then determines the input to the synthesis filter such that the reconstructed signal quality is the best possible, all the while respecting the bit rate constraint (2.4 kb/s). It uses analysis frames of approximately 20 ms and assumes that the signal is locally stationary in each frame. We write as follows:

$$\underline{x} = [x(0) \cdots x(N-1)]^t$$

the N = 160 samples (for a speech signal sampled at f_e = 8 kHz) in each frame.

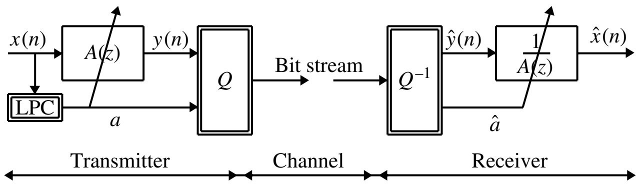

Figure 6.1. *Generalized diagram of a speech coder*

6.2.1. *Determining the filter coefficients*

Determining the filter coefficients is based on the linear prediction theory. Recall that the prediction error power can be expressed as a function of the predictor filter coefficients and the autocovariance function $r_X(k)$ of the process $X(n)$ in the form:

$$\sigma_Y^2 = \sigma_X^2 + 2\underline{r}^t\underline{a} + \underline{a}^t\mathbb{R}\underline{a}$$

where:

$$\underline{a} = \begin{bmatrix} a_1 \\ \vdots \\ a_P \end{bmatrix} \quad \underline{r} = \begin{bmatrix} r_X(1) \\ \vdots \\ r_X(P) \end{bmatrix} \quad \mathbb{R} = \begin{bmatrix} r_X(0) & \cdots & r_X(P-1) \\ \vdots & \ddots & \vdots \\ r_X(P-1) & \cdots & r_X(0) \end{bmatrix}$$

Writing the derivative of σ_Y^2 relative to the vector $\underline{a}$ as equal to the null vector, we obtain the normal equations:

$$\mathbb{R}\underline{a}^{\text{opt}} = -\underline{r}$$

and the minimum power:

$$(\sigma_Y^2)^{\min} = \sigma_X^2 + \underline{r}^t \underline{a}^{\text{opt}}$$

In practice, the autocovariance function needs to be estimated. The statistical mean $r_X(k) = \mathbb{E}\{X(n)X(n-k)\}$ is replaced by the empirical mean:

$$\hat{r}_X(k) = \frac{1}{N} \sum_{n=k}^{N-1} x(n)x(n-k)$$

Recall that the particular choice of summation limits, which means that only the data which lie uniquely in the interval $[0, N-1]$, is owed to the fact that with these limits we obtain a very important practical property: the synthesis filter stability is ensured. Once determined, these coefficients must be quantized, then the code words that allow reconstruction of the synthesis filter coefficients at the receiver must be transmitted.

6.2.2. *Unvoiced sounds*

The linear prediction theory confirms that if $x(n)$ can be considered as the realization of a random process of order P_0, there exists a completely whitening filter $A(z)$ as soon as the order P is greater than or equal to P_0. In this case, we can write the prediction error power spectral density as:

$$S_Y(f) = \sigma_Y^2$$

As we also know that:

$$S_Y(f) = |A(f)|^2 S_X(f)$$

we can deduce that:

$$S_X(f) = \frac{\sigma_Y^2}{|A(f)|^2}$$

Assuming that we choose an ordinary realization of white noise [1] for the synthesis filter input with power $\sigma^2_{\hat{Y}} = \sigma^2_Y$, we can observe the following property at the level of the power spectral densities:

$$S_{\hat{X}}(f) = \frac{\sigma^2_Y}{|A(f)|^2} = S_X(f)$$

The reconstructed signal has the same power distribution as a function of frequency but the waveforms are different. As the ear is relatively insensitive to phase changes, this technique can be used to reconstruct a signal which is perceived to be approximately identical to the original signal. This type of coder is known as a *vocoder*, short for voice coder.

This whole explanation is valid assuming that the speech signal can be considered as the realization of a random process. This hypothesis is quite realistic only for unvoiced sounds.

6.2.3. *Voiced sounds*

The graphs in Figure 6.2 are of a voiced sound in both the time domain (on the left) and the frequency domain (on the right), showing both the original signal $x(n)$ and the prediction error $y(n)$. The filter $A(z)$ is obviously not entirely whitening. A noticeable periodicity remains in the signal $y(n)$, visible in the time domain as well as in the frequency domain. In a 32 ms time interval, there are approximately 7.5 periods. The fundamental frequency is therefore of the order of $f_0 \approx 7.5/0.032 \approx 250$ Hz. We can readily observe a line spectrum in the frequency domain with a fundamental frequency of 250 Hz (corresponding to a female speaker) and the different harmonics.

The problem which now presents itself is to find a model $\hat{y}(n)$ for $y(n)$ that will enable us to obtain $S_{\hat{X}}(f) \approx S_X(f)$ through filtering and which has a very economic bit rate. A comb of the form:

$$\hat{y}(n) = \alpha \sum_{m=-\infty}^{+\infty} \lambda(n - mT_0 + \varphi)$$

is a good candidate. In this expression, $\lambda(n)$ is the Kronecker symbol which takes the value 1 if $n = 0$, 0 otherwise, $T_0 = f_e/f_0$ is the fundamental period expressed as a number of samples, and φ is a value from $\{0, \ldots, T_0 - 1\}$ which conveys the uncertainty in the phase. The signal $\hat{y}(n)$ can perhaps be interpreted as the realization of a random process $\hat{Y}(n)$ whose properties we are now interested in determining.

1. Using the Matlab® function randn, for example.

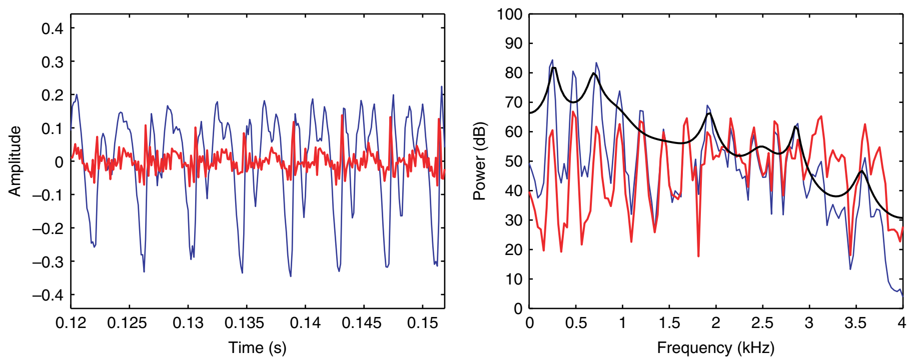

Figure 6.2. *Voice sound. Original signal: fine line; prediction error: thick line*

Note that the mean is expressed as:

$$\mathbb{E}\{\hat{Y}(n)\} = \alpha \sum_{\varphi=0}^{T_0-1} \frac{1}{T_0} \sum_{m=-\infty}^{+\infty} \lambda(n - mT_0 + \varphi) = \frac{\alpha}{T_0}$$

and that the autocorrelation function [2] which is given by:

$$r_{\hat{Y}}(k,n) = \alpha^2\, \mathbb{E}\{ \sum_{m=-\infty}^{+\infty} \lambda(n - mT_0 + \varphi) \sum_{l=-\infty}^{+\infty} \lambda(n - k - lT_0 + \varphi)\}$$

has the value:

$$r_{\hat{Y}}(k,n) = \alpha^2\, \mathbb{E}\left\{ \sum_{m=-\infty}^{+\infty} \lambda(n - mT_0 + \varphi) \right\} = \frac{\alpha^2}{T_0}$$

if k is a multiple of T_0, 0 otherwise. The mean and the autocorrelation function are independent of the observation time n, the process $\hat{Y}(n)$ is stationary with mean $1/T_0$ and the autocorrelation function:

$$r_{\hat{Y}}(k) = \frac{\alpha^2}{T_0} \sum_{m=-\infty}^{+\infty} \lambda(k - mT_0)$$

The power spectral density $S_{\hat{Y}}(f)$ is the discrete Fourier transform of $r_{\hat{Y}}(k)$. We therefore have:

$$S_{\hat{Y}}(f) = \frac{\alpha^2}{T_0} \sum_{k=-\infty}^{+\infty} \sum_{m=-\infty}^{+\infty} \lambda(k - mT_0) \exp(-j2\pi f k)$$

2. It has a much more simple expression than the autocovariance function does.

$$S_{\hat{Y}}(f) = \frac{\alpha^2}{T_0} \sum_{m=-\infty}^{+\infty} \exp(-j2\pi f m T_0)$$

or equivalently:

$$S_{\hat{Y}}(f) = \alpha^2 \sum_{m=-\infty}^{+\infty} \delta\left(f - \frac{m}{T_0}\right)$$

by remembering that a Dirac impulse train leads to a type of Fourier series and we therefore have the general equation:

$$\sum_{n=-\infty}^{+\infty} \delta(t - nT) = \frac{1}{T} \sum_{n=-\infty}^{+\infty} \mathrm{e}^{j2\pi(n/T)t}$$

As the right-hand graph in Figure 6.2 was obtained from a finite number of observed data, we "see" on the principal lobe of the discrete Fourier transform of the weighting window on this graph. In this simulation, a Hamming window with length $2N$ was taken. The principal lobe has a width of $4f_e/2N$ which corresponds to 100 Hz, a value which agrees with that observed in the graph.

The filtering of $\hat{y}(n)$ by a filter with the square of the modulus of the frequency response is represented in Figure 6.2 (spectral envelope) and it yields a signal with the power spectral density which is superimposed on the graph.

6.2.4. *Determining voiced and unvoiced sounds*

The LPC-10 coder must determine whether the current analysis frame contains a voiced or an unvoiced sound. This is performed by realizing an estimate of the autocorrelation function normalized by the prediction error $y(n)$ for numerous values of k. If this function decreases quickly toward 0 (in comparison with a threshold), we can deduce that the sound is unvoiced. In the case of a voiced sound, this function is almost periodic with period T_0 which provides an estimate for the fundamental period. In practice, some precautions are required because a rudimentary algorithm tends to determine the second or even the third harmonic as the fundamental frequency. In practice, we need to ensure that transitions from one analysis frame to another are not too abrupt. In particular, we need to ensure that the choice of φ (uncoded information) has a soft transition between consecutive combs.

6.2.5. *Bit rate constraint*

For every 20 ms, the following parameters are transmitted:

– The coefficients for the filter $A(z)$. In the LPC-10 coder, $P = 10$. Taking approximately 3 or 4 bits for a code coefficient, we find a bit rate of the order of 1.8 kb/s.

– The power σ_Y^2 in the case of an unvoiced sound or in the same way for the coefficient α in the case of a voiced sound which costs 50 $\times$ 6 = 300 bit/s (to cover 50 dB in 1 dB steps).

– The voiced/unvoiced distinction is 50 bit/s.

– Finally, the fundamental period $50 \times \log_2(T_0^{\max} - T_0^{\min})$ = 350 bits/s.

Summing up all these contributions, an overall bit rate of 2.5 kb/s is obtained!

6.3. The CELP coder

6.3.1. *Introduction*

Most telephone band speech coders, especially for bit rates in the range 4.8–16 kb/s, are *Code Excited Linear Predictive* (CELP) coders. The operating principle of these coders was introduced by Atal in the beginning of the 1980s.

Atal's first basic idea [ATA 82] was to propose a new model for excitation of the synthesis filter in the form:

$$\hat{y}(n) = \sum_{k=1}^{K} g_k \lambda(n - n_k)$$

and to calculate the g_k and n_k parameters of this model while minimizing the criterion:

$$\sum_{n=0}^{N-1} [x(n) - \hat{x}(n)]^2$$

There are two new features in this method when compared with determining the input for the LCP-10 coder synthesis filter for voiced sounds. We are no longer dealing with a comb because the impulse positions are no longer regular, and we no longer render $S_{\hat{X}}(f) \approx S_X(f)$ but instead $\hat{x}(n) \approx x(n)$. This is known as modeling *by synthesis*. At the transmitter, the output that will be realized at the receiver is constructed explicitly.

The second innovation by Atal, three years later [SCH 85], was to make use of vector quantization which had just been developed. This idea involves suggesting an

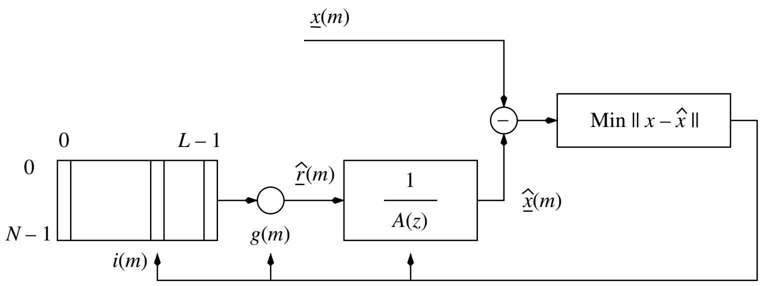

Figure 6.3. *Principle of the CELP coder*

N-dimensional vector, chosen from a predefined codebook, as a possible candidate for the synthesis filter input, as shown in Figure 6.3, and a gain g.

The diagram for the principle of this technique shows that the problem can be expressed simply. Knowing the N samples of the current analysis frame $\underline{x} = [x(0) \cdots x(N-1)]$, we need to determine the coefficients $a_1 \cdots a_P$ of an autoregressive filter of order P, the number i of a vector from the codebook, and the gain g that minimize the error vector norm. The joint minimization is difficult, so it is done in two steps. In the first step, we assume that the filter receives a white input. We therefore attempt to model speech signals as an autoregressive random process. We perform an *LPC analysis*. Once the coefficients are determined, we then need to characterize the filter input.

Another schematic diagram, exactly equivalent to the previous one, can be produced and is shown in Figure 6.4, making use of "information theory" rather than a "parametric modeling" approach and some comments can be made as follows.

This method is in principle a gain-shape vector quantization. We have already seen in the context of the course on vector quantization that minimizing $||\underline{x} - \hat{g}^j \hat{\underline{x}}^k||$ relative to j and k rather than $||\underline{x} - \hat{\underline{x}}^k||$ relative to k uniquely allows us to manage the instantaneous signal power as it evolves through time. This works on the fact that the characteristic form of the signal's spectral content (a normalized vector) and the gain (a scalar) that characterizes the signal power are relatively uncorrelated.

Next, a regular adaptation of the vector quantization codebook to the local signal statistics is realized through a filtering operation. A production model is used which updates every 20 ms for every $N = 160$ samples. Two levels of codebooks are used. The first codebook, composed of the vectors $\{\underline{c}^0 \cdots \underline{c}^{L-1}\}$, is generally chosen in such a way that the space R^N is more or less uniformly covered. It is fixed and is known in identical terms at the transmitter and receiver. The second codebook is updated regularly by filtering. The filter parameters are calculated by realizing an LPC analysis

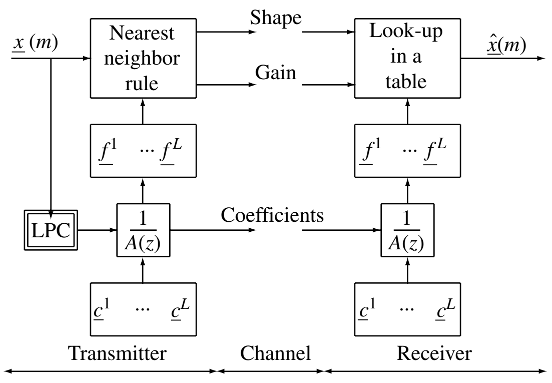

Figure 6.4. *Alternative CELP coder principle diagram*

from N samples of the signal in the current analysis frame. This filtering operation aims to partition the space R^N finely where the vector $\underline{x}$ has a strong probability of appearing and coarsely elsewhere.

Two other characteristics are introduced in this chapter. The first is the use of a weighting function in the frequency domain so that the quantization noise is put into shape. This shaping of the noise is very different from the case of music coders where hearing models are used as we will see later on because, for the target bit rates (4.8–16 kb/s), the quantization noise is too high and the useful band (less than 4 kHz), is far too reduced to justify a precise model. The second characteristic (not shown in the diagram) is the need to introduce a *long-term predictor* which has an ADPCM structure.

6.3.2. *Determining the synthesis filter coefficients*

The first process involves determining and then coding the coefficients used in the synthesis filter. Linear predictive modeling is quasi-systematically used. A compromise is made in choosing the prediction order P. P must be sufficiently high to reproduce the speech spectrum correctly: an order of 8 is necessary to create four peaks in the spectrum and we know that speech signals generally have four formants. However, the prediction order must be as low as possible to economize the bit rate. Therefore, a value between 8 and 16 is usually chosen for P.

The filter coefficients must be coded. In practice, the coefficients $a_1 \cdots a_P$ of the filter $A(z)$ are not quantized directly because they have poor properties for coding, their dynamics are especially poorly controlled. Quantization, whether scalar or vector, of the coefficients $k_1 \cdots k_P$ of the corresponding trellis filter is preferred.

These coefficients can be calculated from normalized correlation coefficients $\rho_1 \cdots \rho_P$ by using the Levinson algorithm. They have the useful property of always being between -1 and $+1$ (if the filter is stable), but the statistical distribution of the coefficients is not uniform at all in this interval. As it is more straightforward to quantize a scalar that has a relatively uniform distribution, these coefficients can be subjected to a non-linear transform of the form:

$$K_i = \log \frac{1 + k_i}{1 - k_i}$$

In effect, we know that when the poles of the transfer function $1/A(z)$ approach the unit circle or, similarly, when the k_i takes values close to 1, the frequency response $1/A(f)$ is at an ever sharper maximum. These new coefficients are known as the *Log Area Ratios*.

There is a fourth representation of these coefficients, equivalent to the three previous representations, which has even better properties for coding. This is the *Line Spectral Pairs* (LSP) representation [ITA 75]. A very brief presentation is given here. From a polynomial of order P:

$$A(z) = 1 + \sum_{i=1}^{P} a_i z^{-1}$$

two new polynomials of order $P+1$ are constructed:

$$B_1(z) = A(z) + z^{-P-1} A(z^{-1})$$

$$B_2(z) = A(z) - z^{-P-1} A(z^{-1})$$

We can show that the two polynomials have the following properties:

– $B_1(z)$ is a palindromic polynomial, whereas $B_2(z)$ is an anti-palindromic polynomial.

– If all the roots of $A(z)$ are inside the unit circle, all the roots of $B_1(z)$ and $B_2(z)$ are on the unit circle.

– The roots of $B_1(z)$ and $B_2(z)$ appear alternately on the unit circle.

– If P is even (usual case), we can write $B_1(z)$ and $B_2(z)$ in the form:

$$B_{1,2}(z) = (1 \pm z^{-1}) \prod_{i=1}^{P/2} (1 - 2\cos\phi_{2i-1} z^{-1} + z^{-2})$$

– The LSP $\phi_1 \cdots \phi_P$ satisfy:

$$0 < \phi_1 < \cdots < \phi_P < \pi$$

Knowing the coefficients a_i, the coefficients ϕ_i can be deduced. The reciprocal is immediate because $A(z) = (B_1(z) + B_2(z))/2$. Quantizing the coefficients ϕ_i must preserve the inequalities above to maintain the synthesis filter stability. If the difference between two consecutive coefficients is quantized, the relations are automatically preserved. In Figure 6.5, the polynomial roots are shown for $A(z)$ (crosses), $B_1(z)$ (circles), and $B_2(z)$ (squares) for an LPC analysis of order $P = 12$ on the voiced sound shown in Figure 6.2. The alternating roots for $B_1(z)$ and $B_2(z)$ can be clearly seen on the unit circle.

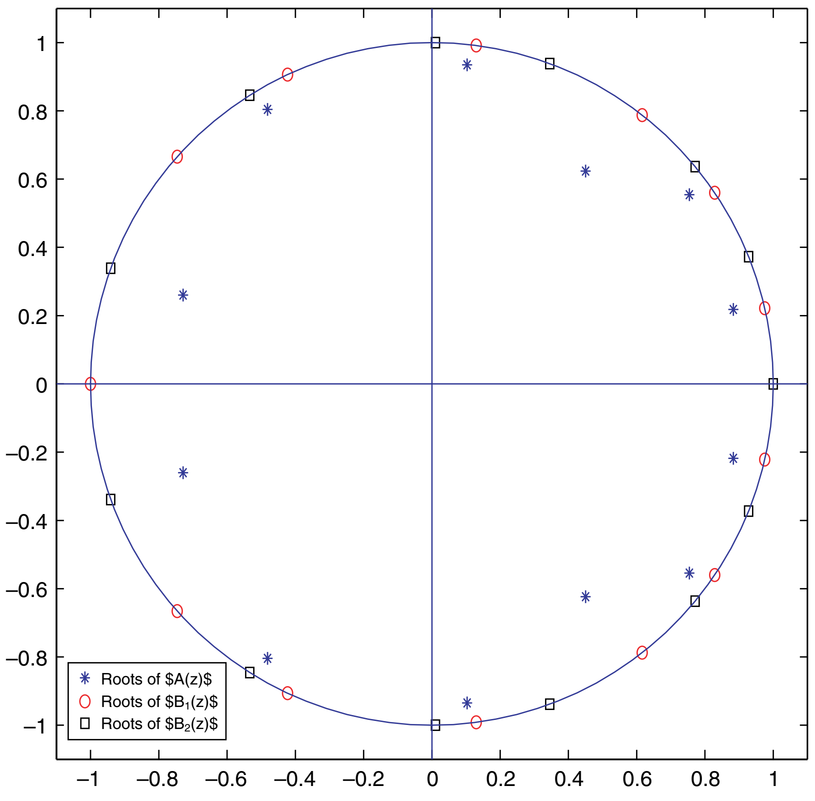

Figure 6.5. *LSP coefficients: roots of polynomials $A(z)$, $B_1(z)$, and $B_2(z)$*

6.3.3. *Modeling the excitation*

6.3.3.1. *Introducing a perceptual factor*

The quadratic cost functions lend themselves well to calculations: they have the useful property of producing a linear system when this criterion is derived with regard to the unknown parameters. However, this criteria is not particularly well adapted to our hearing system. A perception correction is very widely used to alleviate this inconvenience [ATA 79]. A weighting function, in the form of a filter transfer function $W(z)$, is added before the minimization criterion, as shown in Figure 6.6.

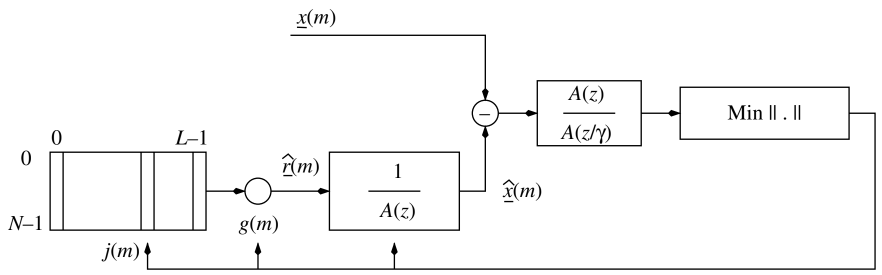

Figure 6.6. *Introducing a weighting function* $W(z) = A(z)/A(z/\gamma)$

A presentation of the phenomena of masking one sound by another will be given later. Let us simply say for the moment that the quantization noise is less perceptible when the signal has plenty of energy. We can say that the signal masks the noise. The total power of the quantization noise cannot be played about with. However, it is possible to alter the spectral shape of the noise. We can therefore look for a weighting function which attributes less importance to energetic frequency regions, that is, to formant regions. We can show that the transfer function $W(z) = A(z)/A(z/\gamma)$ with $0 < \gamma < 1$ can fulfill this role. In effect, if we note:

$$A(z) = 1 + a_1 z^{-1} + \cdots + a_P z^{-P} = \prod_{i=1}^{P}(1 - p_i z^{-1})$$

where p_i specifies the ith root of the polynomial $A(z)$, we observe that:

$$A\left(\frac{z}{\gamma}\right) = 1 + a_1 \gamma z^{-1} + \cdots + a_P \gamma^P z^{-P} = \prod_{i=1}^{P}(1 - \gamma p_i z^{-1})$$

The modulus of the frequency response of the filter $1/A(z/\gamma)$ shows peaks that are less sharp than those of the filter $1/A(z)$ because the poles of the filter $1/A(z/\gamma)$ are brought back toward the center of the unit circle compared with those of the filter $1/A(z)$. The modulus of the frequency response filter $W(z) = A(z)/A(z/\gamma)$ therefore has the desired shape.

The diagram of the modeling principle is shown in Figure 6.7.

This diagram clearly shows the fact that we are attempting to model the *perceptual signal* $\underline{p}$ by $\hat{\underline{p}}$. The filter characterized by the transfer function $1/A(z/\gamma)$ is known as

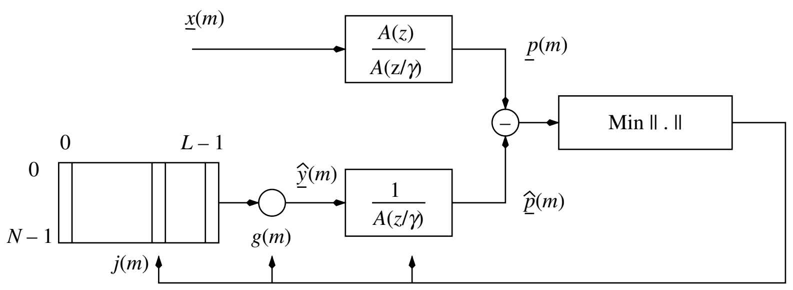

Figure 6.7. *Modeling the perceptual signal*

the *perceptual weighting filter*, or in short the perceptual filter. The numerical value of the perceptual factor γ allows the operator to alter the weighting factor conveniently. When γ = 1, the filter operates as though there is no weighting function in use; the original signal is modeled. The reconstruction error $\underline{x} - \hat{\underline{x}}$ is approximately white in this case. When γ = 0, a modeling of the residual signal is realized. In this case, the reconstruction error has the spectral shape of the original signal. In general, a value close to 0.8 is chosen for γ.

6.3.3.2. *Selecting the excitation model*

The diagram in Figure 6.3 leads us to think that we choose a unique vector from a codebook and a single gain for the excitation model. For greater generality, we use an input in the form:

$$\hat{\underline{y}} = \sum_{k=1}^{K} g_k \underline{c}^{j(k)}$$

where K is the number of vectors in the excitation, also known as the modeling order for the excitation (usually K = 2 or 3). *A priori*, each vector can come from a distinct codebook as we will see shortly, but to begin with we assume that these K vectors come from the same codebook $C = \{\underline{c}^0 \cdots \underline{c}^{L-1}\}$. The problem therefore consists of finding K vectors $c^{j(1)} \cdots c^{j(K)}$ in an excitation codebook and K gains $g_1 \cdots g_K$ such that the vector $\hat{\underline{y}}$, filtered by the perceptual filter $1/A(z/\gamma)$, gives rise to the modeled vector $\hat{\underline{p}}$ which resembles the vector $\underline{p}$ as closely as possible.

6.3.3.3. *Filtered codebook*

Having fixed the form of the excitation model, we now need to determine the numerical values for the parameters. We minimize $||\underline{p} - \hat{\underline{p}}||^2$ relative to the indices $j(1) \cdots j(K)$ and gains $g_1 \cdots g_K$. We therefore need to show what is known in the criterion, that is, what we calculated before and what is explicitly dependent upon the unknown $j(k)$ and g_k. The modeled perceptual signal is expressed as:

$$\hat{p}(n) = \sum_{i=0}^{\infty} h(i)\hat{y}(n-i) \quad \text{for} \;\; n = 0 \cdots N-1$$

where $h(n)$ is the impulse response of the perceptual filter and where $n = 0$ is the conventional characterization of the first sample of the current analysis frame. The previous expression decomposes into two terms:

$$\hat{p}(n) = \sum_{i=0}^{n} h(i)\hat{y}(n-i) + \sum_{i=n+1}^{\infty} h(i)\hat{y}(n-i)$$

The first term is *a priori* unknown, but the second term is known since it involves $\hat{y}(n)$ for $n < 0$, that is, the excitation of the synthesis filter determined in the previous analysis frame. Finally, the modeled perceptual signal is written as:

$$\hat{p}(n) = \sum_{k=1}^{K} g_k \sum_{i=0}^{n} h(i)c^{j(k)}(n-i) + \sum_{i=n+1}^{\infty} h(i)\hat{y}(n-i)$$

We note that:

$$\underline{\hat{p}}^0 = \left[\sum_{i=1}^{\infty} h(i)\hat{y}(-i) \cdots \sum_{i=N}^{\infty} h(i)\hat{y}(N-1-i)\right]^t$$

is the contribution in the current frame of the excitation which comes from previous frames. This is called the *ringing* vector, perhaps because a bad production of this term results in an artifact of frequency f_e/N which is unpleasant to hear. We note:

$$\underline{f}^j = [f^j(0) \cdots f^j(N-1)]^t$$

where:

$$f^j(n) = \sum_{i=0}^{n} h(i)c^j(n-i)$$

the result of filtering the vector $\underline{c}^j$ by the perceptual filter from zero initial conditions. This filtering operation can be characterized by the matrix expression:

$$\underline{f}^j = \begin{bmatrix} h(0) & 0 & \cdots & 0 \\ h(1) & \ddots & \ddots & \vdots \\ \vdots & \ddots & \ddots & 0 \\ h(N-1) & \cdots & h(1) & h(0) \end{bmatrix} \underline{c}^j = H\underline{c}^j$$

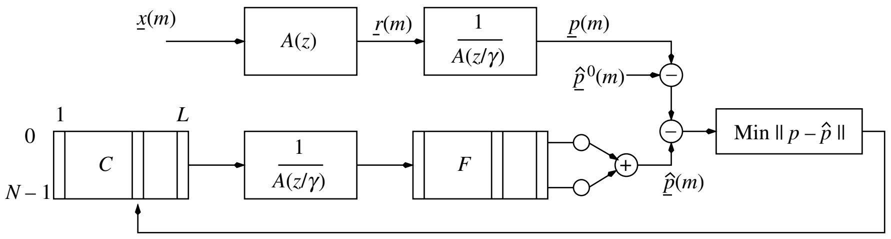

Figure 6.8. *Modeling the perceptual signal by K vectors from the filtered codebook*

The new diagram of the principle is now shown in Figure 6.8. The excitation codebook C and the filtered codebook F are, both of them comprising L vectors, denoted by $\underline{c}^j$, and $\underline{f}^j$, respectively, as shown in Figure 6.8.

We can make the following remark on this result. In all the previous expressions, we have assumed, implicitly, that the filters in use were invariant. In fact, they are only locally invariant. The vector $\underline{\hat{p}}^0$ is therefore the free response of the filter of which the coefficients are characteristic of the current frame. The state vector of the filter is not affected when we pass from one frame to another.

6.3.3.4. *Least squares minimization*

From now onwards, to simplify the notation, $\underline{p}$ is the perceptual vector to which the contribution from the preceding frames is subtracted. The problem of determining the excitation in a CELP coder can be expressed as follows. Knowing $\underline{p}$ and $\underline{f}^0 \cdots \underline{f}^{L-1}$, find indices $j(1) \cdots j(K)$ and gains $g_1 \cdots g_K$ to minimize:

$$D = ||\underline{p} - \underline{\hat{p}}||^2 = ||\underline{p} - \sum_{k=1}^{K} g_k \underline{f}^{j(k)}||^2$$

In matrix notation, the problem can be re-expressed: given a matrix F belonging to R^{N*L}, composed of L column vectors $\underline{f}^j$ and a vector $\underline{p}$ belonging to R^N, extract from F a matrix A belonging to R^{N*K}, composed of K column vectors from L, and find a K-dimensional vector $\underline{g}$ to minimize:

$$D = ||\underline{p} - A\underline{g}||^2 = ||\underline{p} - \underline{\hat{p}}||^2$$

This is a classic minimization problem in the least squares sense if we assume that the indices $j(1) \cdots j(K)$, that is, the matrix A is known [GOL 89]. $\underline{g}$ is determined from the overdetermined system $\underline{p} = A\underline{g}$. We can write that the best approximation $\underline{\hat{p}}$ of $\underline{p}$ is the orthogonal projection of $\underline{p}$ in the subspace created by the column vectors of A or that the vector $\underline{p} - \underline{\hat{p}}$ is orthogonal to all the vectors created in this subspace. We obtain the *normal equations*:

$$(\underline{f}^{j(k)})^t(\underline{p} - A\underline{g}) = 0 \quad \text{for } k = 1 \cdots K$$

or in matrix form:

$$A^t A\underline{g} = A^t \underline{p}$$

The matrix A generally has full rank. The matrix $A^t A$ is therefore positive-definite. This resolution makes use of the Choleski decomposition, for example, or by rapid algorithms if we set a Toeplitz form or near-Toeplitz form for this matrix. Unfortunately, the indices are unknown and must be determined simultaneously with the gains.

There is an optimum algorithm: we need to determine on the matrices A which correspond to all the combinations of k vectors from L, to resolve the normal equations for each matrix A, to evaluate the criterion, and to select the combination that minimizes this criterion. Knowing that $L!/K!(L-K)!$ possible combinations exist, and that typical values for K, L, and N are 3, 256, and 40, respectively, we have $256 * 255 * 254/6$, that is, almost 3 million linear systems of order 3, to resolve every 5 ms! We must therefore restrict ourselves to a sub-optimal algorithm.

Let us specify the problem precisely. Minimizing D involves choosing a combination of $j(1) \cdots j(K)$, solving the normal equations $A^t A\underline{g} = A^t \underline{p}$, and finally selecting the combination that maximizes $\underline{g}^t A^t A\underline{g}$ because minimizing $||\underline{p} - \hat{\underline{p}}||^2$ is equivalent to maximizing $||\hat{\underline{p}}||^2$. In effect, it is enough to apply Pythagoras' theorem:

$$||\underline{p} - \hat{\underline{p}}||^2 = ||\underline{p}||^2 - ||\hat{\underline{p}}||^2$$

and note that $||\underline{p}||^2$ is constant. Because:

$$\underline{g} = (A^t A)^{-1} A^t \underline{p}$$

the problem finally involves selecting the combination $j(1) \cdots j(K)$ that maximizes:

$$||\hat{\underline{p}}||^2 = \underline{p}^t A (A^t A)^{-1} A^t \underline{p}$$

and then to calculate the gains to solve, once only, the normal equations which correspond to the chosen combination.

6.3.3.5. *Standard iterative algorithm*

The difficulty arises essentially from inverting the matrix $A^t A$. There are two ways (at least) of simplifying the process. By limiting the search to one vector at a time, the expression $A^t A$ is a scalar, and calculating the indices and gains can be done iteratively. Otherwise, specific properties of the matrix A can be set, for example, $A^t A = I$ whatever the combination may be. The first method is rather standard. It was proposed by [ATA 82] for a multi-impulse excitation. It is widely used in practice but it is, unfortunately, sub-optimal. The second method is an optimum algorithm but imposes very strong restrictions on the matrix F: it must be composed of orthogonal vectors.

We present here the standard iterative algorithm. We denote by $< \underline{x}, \underline{y} >$ the scalar product of the two vectors $\underline{x}$ and $\underline{y}$. In the first iteration, a single vector $\underline{f}^j$ is selected. We have:

$$A^t A =< \underline{f}^j, \underline{f}^j >$$

$$A^t \underline{p} =< \underline{f}^j, \underline{p} >$$

We must therefore choose the index j that maximizes:

$$< \underline{p}, \underline{f}^j >< \underline{f}^j, \underline{f}^j >^{-1}< \underline{f}^j, \underline{p} >= \frac{< \underline{f}^j, \underline{p} >^2}{< \underline{f}^j, \underline{f}^j >}$$

then calculate the gain:

$$g_1 = \frac{< \underline{f}^{j(1)}, \underline{p} >}{< \underline{f}^{j(1)}, \underline{f}^{j(1)} >}$$

In the kth iteration, the contribution of the first $k-1$ vectors $\underline{f}^{j(i)}$ is taken off $\underline{p}$:

$$\underline{p}^k = \underline{p} - \sum_{i=1}^{k-1} g_i \underline{f}^{j(i)}$$

and a new index $j(k)$ and a new gain g_k are calculated to satisfy:

$$j(k) = \arg \max_j \frac{< \underline{f}^j, \underline{p}^k >^2}{< \underline{f}^j, \underline{f}^j >}$$

$$g_k = \frac{< \underline{f}^{j(k)}, \underline{p}^k >}{< \underline{f}^{j(k)}, \underline{f}^{j(k)} >}$$

6.3.3.6. *Choosing the excitation codebook*

This algorithm can be applied whatever the excitation codebook contents may be. If we choose $C = I$, where C is the matrix composed of column vectors $\underline{c}^j$ and I is the $N * N$-dimensional identity matrix, we obtain a "multi-impulse" excitation. The selected indices $j(k)$ characterize the chosen impulse positions and the gains g_k define the amplitudes. In the standard CELP coder, the excitation codebook is initialized with values generated from zero-mean Gaussian random variables or constructed through training using a variation of the Lloyd–Max algorithm. In these two cases, the complexity of the process is quite substantial. To reduce the complexity, we can define a codebook which has a strong structure, for example, by setting ternary values $\{+1, -1, 0\}$ and choosing regular codebook positions for non-zero values. This is the way that the G.729 coder [UNI 96] works, and for this reason it is called the ACELP coder (*Algebraic CELP*).

6.3.3.7. *Introducing an adaptive codebook*

We know that to determine coefficients for the filter $A(z)$ we minimize the prediction error energy:

$$D_1 = \sum_n [x(n) - \sum_{i=1}^{P} a_i x(n-i)]^2$$

with respect to P unknown parameters a_i. This is known as short-term prediction because, to predict the value of the signal at time n, we use the P previous samples. Once the calculation has been done directly by solving the linear system obtained by deriving D_1 with respect to the P unknown parameters, the signal $x(n)$ is filtered by the filter $A(z)$ with order P. The short-term residual signal $y(n)$ is obtained. The display of the signal graphically, especially in the case of voiced sounds, shows that not all the redundancy in the speech signal has been removed. A certain periodicity remains as shown in Figure 6.2. This periodicity corresponds, physiologically, to the vibration period of the vocal cords. We can characterize this information by introducing two new parameters b and Q then minimizing the energy of a new prediction error:

$$D_2 = \sum_n [y(n) - by(n-Q)]^2$$

with respect to these two unknown parameters. This is known as long-term prediction. We can remark that this minimization cannot by realized in the same way as the previous one since for D_1, P is fixed, whereas for D_2, Q is a parameter to be determined.

Open-loop minimization

To calculate the optimum values of B and Q, we need to only derive D_2 with respect to b to obtain the optimum b as a function of Q, carry this value forward into D_2 then choose a value for Q which minimizes the criterion. This is not strictly the best adapted solution. In general, a closed-loop solution is preferred.

Closed-loop minimization

Let us introduce the transfer function:

$$B(z) = 1 - bz^{-Q}$$

In the synthesis stage, the inverse filter $1/B(z)$ is used which must be placed upstream from the filter $1/A(z)$. The diagram that corresponds to this function is given in Figure 6.9.

Let us revisit the previous exposition, this time assuming that b and Q are predetermined. We need to find the vectors $\underline{c}^{j(k)}$ in the excitation codebook and the

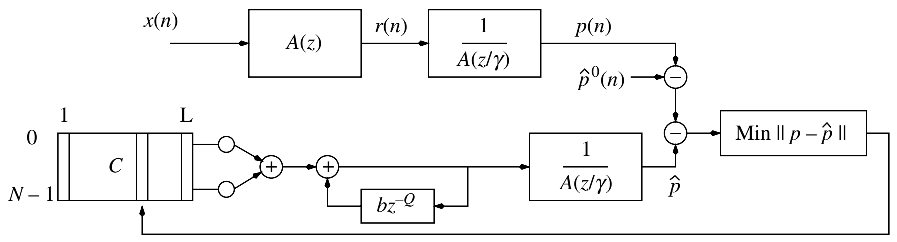

Figure 6.9. *Short-term and long-term prediction*

gains g_k so that the vector $\sum_{k=1}^{K} g_k \underline{c}^{j(k)}$, when filtered by the filter $1/B(z)$ and then the perceptual filter $1/A(z/\gamma)$, results in the modeled vector $\underline{\hat{p}}$ which is the closest possible resemblance of the vector $\underline{p}$. We have seen that the modeled perceptual signal is written as:

$$\hat{p}(n) = \sum_{i=0}^{n} h(i)\hat{y}(n-i) + \sum_{i=n+1}^{\infty} h(i)\hat{y}(n-i) \quad \text{for} \;\; n = 0 \cdots N-1$$

But $\hat{y}(n)$ for $n \geq 0$, which is *a priori* unknown, is composed of an unknown part which depends only on $\underline{c}^{j(k)}$ and g_k and a known part:

$$\hat{y}(n) = \sum_{k=1}^{K} g_k c^{j(k)}(n) + b\hat{y}(n-Q)$$

if we allow the hypothesis that the long-term predictor parameters b and Q have been determined and that:

$$n - Q < 0 \quad \forall n \in 0 \cdots N-1$$

that is:

$$Q \geq N$$

The delay Q must therefore be greater than or equal to the analysis frame size. Finally, the modeled perceptual signal is written as:

$$\hat{p}(n) = \sum_{k=1}^{K} g_k \sum_{i=0}^{n} h(i)c^{j(k)}(n-i) + b\sum_{i=0}^{n} h(i)\hat{y}(n-i-Q) + \sum_{i=n+1}^{\infty} h(i)\hat{y}(n-i)$$

We can write:

$$\underline{\hat{p}}^k = [\hat{p}^k(0) \cdots \hat{p}^k(N-1)]^t$$

with:

$$\hat{p}^0(n) = \sum_{i=n+1}^{\infty} h(i)\hat{y}(n-i)$$

$$\hat{p}^1(n) = b\sum_{i=0}^{n} h(i)\hat{y}(n-i-Q)$$

$$\hat{p}^k(n) = g_k\sum_{i=0}^{n} h(i)c^{j(k)}(n-i)$$

We notice that $\underline{\hat{p}}^1$ and $\underline{\hat{p}}^k$ are in exactly the same form and can be interpreted in the same way, as the filtering of a known vector by the perceptual filter starting from zero initial conditions and weighted by a gain. The standard algorithm therefore consists first of all of minimizing the criterion:

$$||(\underline{p} - \underline{\hat{p}}^0) - \underline{\hat{p}}^1||^2$$

to determine Q and b then to minimize the criterion:

$$||(\underline{p} - \underline{\hat{p}}^0 - \underline{\hat{p}}^1) - \underline{\hat{p}}^2||^2$$

to determine j(2) g_2, etc. Note that the two long-term predictor parameters Q and b can be calculated in exactly the same way as the parameters $j(k)$ and g_k with respect to the condition for constructing an *adaptive* codebook which involves the past excitation:

$$C = \begin{bmatrix} \hat{y}(-Q_{\max}) & \cdots & \hat{y}(-2N) & \hat{y}(-2N+1) & \cdots & \hat{y}(-N) \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ \hat{y}(-Q_{\max}+N-1) & \cdots & \hat{y}(-N-1) & \hat{y}(-N) & \cdots & \hat{y}(-1) \end{bmatrix}$$

This codebook has two interesting properties. First, the associated matrix has a Toeplitz structure. We can see that this property allows us to reduce the number of operations undertaken when the codebook is filtered. Second, when we pass from one analysis frame to another, the entire codebook is not called into question. Only N vectors must be updated. The others are deduced by translation leftward.

Remarks

The constraint $Q \geq N$ is too strong in practice. It is necessary to introduce a sub-frame processing and to determine an excitation model for each sub-frame. In effect, the mean fundamental frequency is of the order of 100 Hz for a male speaker and 250 Hz for a female speaker. This means that the likely values for Q are 80 and 32. Since the value usually chosen for N is 160, it seems necessary to divide the analysis frame into, at least, five sub-frames but generally this is limited to four sub-frames for $N^{'}$ = 40 samples since more sub-frames would become too costly in terms of bit rate.

The codebook C can be extended to the right by constructing it in the following way. The first time, the $N-1$ available samples $\hat{y}(-N+1)\cdots\hat{y}(-1)$ are used and are supplemented by $\hat{y}(-N+1)$; the second time around, $N-2$ samples $\hat{y}(-N+2)\cdots\hat{y}(-1)$ are used, supplemented by $\hat{y}(-N+2)\hat{y}(-N+3)$, etc. We stop at index $Q_{\min}$. This artificially solves the problem of the size of the sub-frame being too high for the fundamental period of a female speaker. Choosing $Q_{\min}$ = 20 is enough, for example. We take:

$$Q_{\max} = Q_{\min} + 127$$

if we decide to code the index on seven bits.

6.3.4. *Conclusion*

We have provided here only the principles based on which CELP speech coders are built. To obtain a reconstructed signal at a suitable quality, it is necessary to govern precisely all the parameters in play and to add a number of tricks. To get an idea of these, we can consult the paper [SAL 98].

We provide here information transmitted in the network in the case of the 8 kb/s G.729 coder. The $P = 10$ coefficients of the synthesis filter $1/A(z)$ are updated for every 10 ms (80 sample analysis frame). The LSP coding is realized with 18 bits. The synthesis filter input is the sum of two signals updated for every 5 ms. The first corresponds to the long-term predictor. The parameter Q and the gain are coded on 7 and 3 bits, respectively. The second comes from a vector quantization codebook. The gain is coded on 4 bits and the index on 13+4 bits, conforming to what is indicated in Table 6.1. Every 10 ms, $18 + 2 \times (3 + 7 + 4 + 17) = 80$ bits must be transmitted, requiring a bit rate of 8 kb/s.

Impulses	Amplitude	Positions	Bits
0	± 1	0, 5, 10, 15, 20, 25, 30, 35	1 + 3
1	± 1	1, 6, 11, 16, 21, 26, 31, 36	1 + 3
2	± 1	2, 7, 12, 17, 22, 27, 32, 37	1 + 3
3	± 1	3, 8, 13, 18, 23, 28, 33, 38 4, 9, 14, 19, 24, 29, 34, 39	1 + 4

Table 6.1. *Every 40 samples of the G.729 coder select 4 "impulses" for which the possible positions are indicated*

We can also provide a few pieces of information on the 3GPP AMR-WB coder (*Adaptive Multi-Rate Wideband* ITU-T G.722.2) [BES 02] which is the first coder adapted for fixed and mobile networks. It removes the requirement for transcoding. As its name indicates, several bit rates are possible. These vary between 6.6 and

23.85 kb/s. The fact that it is wideband noticeably improves the quality. The introduction of bass frequencies 50–200 Hz makes the voice sound more natural and improves the effect of presence. The extension 3.4–7 kHz leads to a greater intelligibility. This is an ACELP coder which is very comparable with the G.729 but it has been subjected to a perceptual filter (extended to wide band) and a modification of the use of pitch information since speech does not have harmonic structure across the whole band 50 Hz–7 kHz, only between 50 Hz and 2 kHz. The main novelty of this coder is the introduction of a very large excitation codebook ($\log_2 L = 88$ bits).

Chapter 7

Audio Coding

7.1. Principles of "perceptual coders"

We show, in Figure 7.1, the amplitude of a violin signal as a function of time (Figure 7.1(a)) and the repartitioning of the signal power as a function of frequency (Figure 7.1(b)).

As for speech coders, an audio coding algorithm is essentially a loop which consists first of filling a buffer with N samples as shown in Figure 7.2, processing these N samples and then passing on to the next analysis frame. The analysis frames always overlap; the shift between two analysis frames is characterized by the parameter $M < N$. The vector $\underline{x}(m)$ is, therefore, of the form:

$$\underline{x}(m) = [x(mM), \cdots, x(mM + N - 1)]^t \otimes [v(0), \cdots, v(N-1)]^t$$

where the operator $\otimes$ represents a multiplication of two vectors component by component and where $\underline{v}$ is a weighting window. The analysis frames are generally around 20 ms in length, the parameter N which must therefore be $44.1 \times 20 = 882$ is equal to 512 (MPEG-1 and AC3) or 2048 (MPEG-2 AAC). The value of the parameter M depends on the coder: $M = 32 \ll N/2$ for the MPEG-1 coder, $M = N/2$ for the AC3 and MPEG-2 AAC coders.

The diagram of a perceptual coder shows that it is composed of three distinct modules which are activated at each analysis frame: a time-frequency transform of the form $\underline{X}(m) = H\underline{x}(m)$, a bit allocation controlled by a hearing model, and a scalar or vector quantization of components of the vector $\underline{X}(m)$ followed by entropy coding. The bit stream transmitted in the channel is made up of code words from the entropy coding as well as side information, for example the result of the bit allocation,

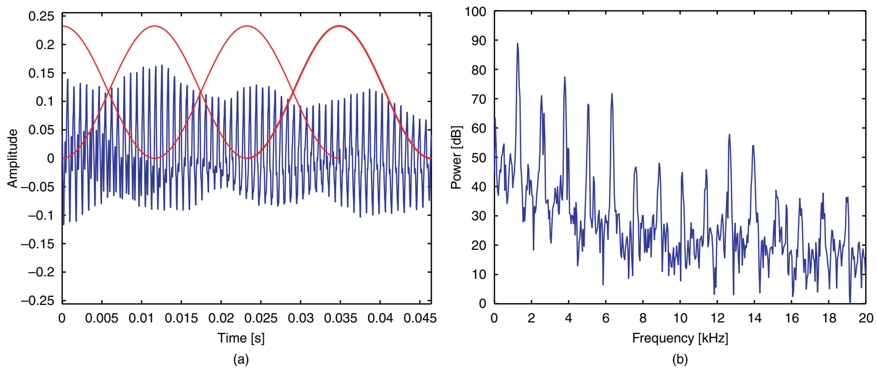

Figure 7.1. *Violin signal in (a) the time domain and (b) the frequency domain*

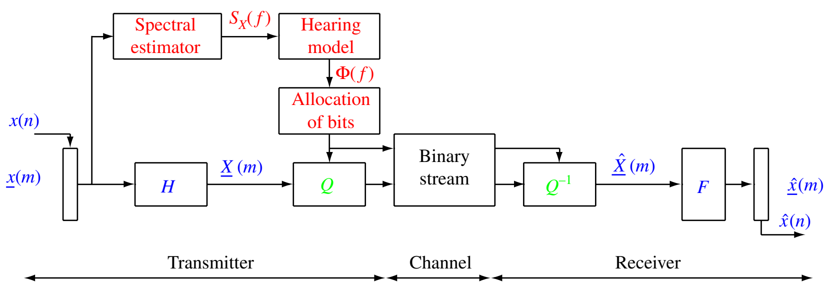

Figure 7.2. *Diagram of a perceptual coder*

which is necessary at the receiver *a priori*. At the receiver, the vector $\hat{\underline{X}}(m)$ must be constructed and brought back into the time domain by the inverse transform $\hat{\underline{x}}(m) = F\hat{\underline{X}}(m)$ then the samples $\hat{x}(n)$ must be extracted at the speed f_e. In the MPEG-2 AAC coder, the reconstruction error $q(n) = x(n) - \hat{x}(n)$ is also calculated at the transmitter: we have a *local copy* of the receiver at the transmitter. This information is used in the module for bit allocation. This is known as *closed loop* coding, as for speech coders.

Recall that in section 3.2 we showed the equivalence of filter banks and transformations. The moduli of the components of the vector $\underline{X}(m)$ are expressed in terms of frequency if the transform has been correctly chosen; in general this is the MDCT transform. We can say, in this introductory section, that these moduli provide a good estimation of the music signal's power spectral density $S_X(f)$ in the current analysis frame.

Perceptual coders aim to eliminate components of the vector $\underline{X}(m)$ which are inaudible and to quantize the remaining components with the minimum number of bits. The role of the hearing model is to define the inaudibility criterion. The psychoacoustic results, shown in section 7.5, show that a pure sound (a sinusoidal waveform) or a narrow-band noise can be made inaudible (masked) by the presence of another pure sound or narrow-band noise. In the graph of Figure 7.3(a), a component of the power spectral density $S_X(f)$ has been isolated. The vaguely peaked curve shows the power of the masked sinusoid just at the limit of inaudibility. Since, in reality, the N samples of a musical sound in the current analysis frame are expressed in the form of a sum of M frequency components, the psychoacoustic results must be generalized and the different contributions added together. We obtain the *masking threshold* $\Phi(f)$ which can be seen superposed on the right-hand graphs with the power spectral density $S_X(f)$.

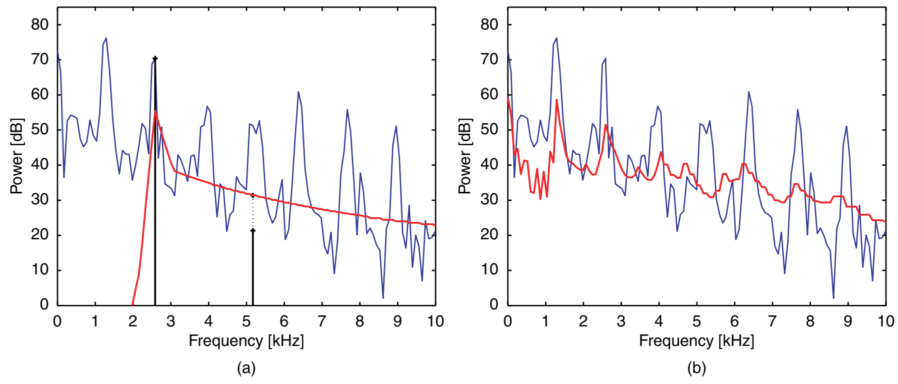

Figure 7.3. *(a) Masking curve (in bold). (b) Masking threshold (in bold). Only contributions in the band* $[0–10]$ *kHz are shown*

The error from a quantizing operation results in an inaudible perturbation if its power spectral density $S_Q(f)$ satisfies the relation:

$$S_Q(f) < \Phi(f)$$

for any f and any analysis frame. Equation [4.17], for which we recall the expression:

$$b \geq \frac{1}{2} \int_{-1/2}^{+1/2} \max \left[0, \log_2 \frac{S_X(f)}{S_Q(f)} \right] \mathrm{d}f$$

gives the necessary bit rate which is enough to code a source with power spectral density $S_X(f)$ with a power spectral density distortion $S_Q(f)$. Since $[S_Q(f)]_{\max} = \Phi(f)$, the frequency axis must be finely partitioned, frequency bands which satisfy

$S_X(f) > \Phi(f)$ must be determined, and the bits must be allocated as a function of the ratio $S_X(f)/\Phi(f)$ by applying the 6 dB per bit rule. The graph in Figure 7.4 shows that, in this analysis frame, approximately 50% of the components can be eliminated and that the mean *signal-to-mask ratio* is of the order of 12 dB which therefore requires 2 bits on average per component. The compression rate is therefore of the order of $2 \times 16/2 = 16$. For music signals in CD format, the lowest bit rate which respects the inaudibility constraint (in this frame) is of the order of 40 kbit/s. We can see immediately that everything depends upon the precision with which the psychoacoustic model is defined. Defining a good model is very tricky.

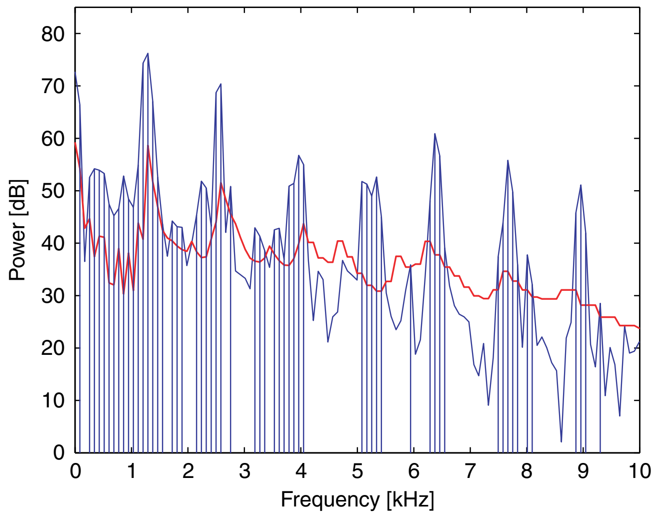

Figure 7.4. *Eliminating masked components: only components of $S_X(f)$ greater than the masking threshold are coded*

In reality, defining an audio coder does not stop at this point. A whole series of specific problems must be studied. The two sections which follow describe the two basic perceptual coders which show that the creation of an audio coder with good performance is the result of numerous subtle compromises. If a more detailed explanation is required; see [PAI 00].

7.2. MPEG-1 layer 1 coder

A few succinct pieces of information are given here pertaining to the audio part of the international standard ISO/CEI 11172 [NOR 93]. This standard authorizes three different sampling frequencies: 32, 44.1 and 48 kHz. It also authorizes a wide range of bit rates. In this section, we assume that the sampling frequency is 44.1 kHz and the target bit rates are between 64 and 96 kbit/s per channel. All the tables provided here are therefore only relevant for this sample frequency and these bit rates.

7.2.1. *Time/frequency transform*

The time/frequency transform used is a bank of $M = 32$ pseudo-quadrature mirror filters (QMF) filters which realize a uniform partition of the frequency axis as shown on the graph in Figure 7.5, which shows only the frequency response of the first filters in the bank. This filter bank is not a perfectly reconstructed one but, in the absence of quantization, the SNR is greater than 90 dB, which is sufficient since the real SNR in CD format is itself of this size.

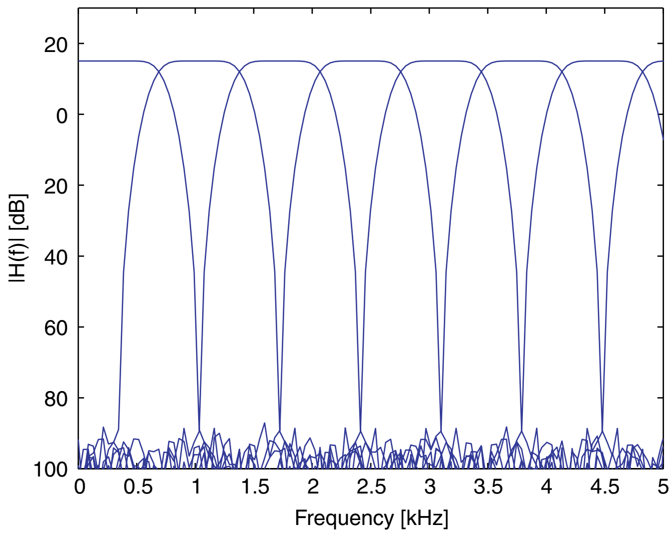

Figure 7.5. *Frequency responses in the PQMF filter bank in the band [0–5] kHz*

This filter bank produces M sub-band signals which are downsampled by a factor of 32. We find the sub-band samples [1] $y_k(m)$. In each sub-band independently, the coder constructs a vector $\underline{y}_k$ which groups 12 samples every $32 \times 12 = 384$ samples for, $x(n)$ which corresponds approximately to around 10 ms. For each vector $\underline{y}_k$, the component with the largest absolute value of all 12 is used to define a *scale factor* g_k

1. To show clearly that these are signals with vector components that can be interpreted as frequencies, we have kept the notation used in section 3.2 rather than the notation $\underline{X}(m)$ from this chapter.

and from this a normalized vector $\underline{a}_k$ is deduced. Each scale factor is expressed in dB then quantized with help of a codebook covering more than 100 dB in steps of 2 dB which requires 6 bits.

The choice of 12 samples is the result of a compromise. For low values, the bit rate associated with scale factors becomes too large. For higher values, echo phenomena become audible since there is no longer any temporal masking. *A priori*, this coder delivers a good temporal resolution (sub-band signals are frequently updated) but a bad frequency resolution (the bank filters bandwidths are of the order of 22/32 = 0.7 kHz).

7.2.2. *Psychoacoustic modeling and bit allocation*

The samples of the signal $x(n)$ contributing to determining the M vectors $\underline{a}_k$ are also used to realize an estimate of the power spectral density $\hat{S}_X(f)$ in the current analysis frame. From $\hat{S}_X(f)$, the coder calculates a masking threshold $\Phi(f)$ using a psychoacoustic model and then a *signal-to-mask ratio* $\hat{S}_X(f)/\Phi(f)$. From the M signal-to-mask ratios, the coder realizes a bit allocation, that is, it determines the number of bits b_k with which each component of the vector $\underline{a}_k$ is quantized. The algorithm used is the standard *greedy* algorithm.

7.2.3. *Quantization*

Next, the quantization of each of the components of these normalized vectors $\underline{a}_k$ is realized using uniform scalar quantizers in the range $[-1, +1]$ with quantization steps function of b_k.

Each sub-band signal has a certain number of possible quantizers. Each quantizer is characterized by a number L of quantization steps and a signal-to-noise ratio σ_X^2/σ_Q^2 (a point in the rate–distortion plane). The values adopted in the ISO standard for the signal-to-noise ratio as a function of the number of quantization steps are given in Table 7.1. For the purposes of comparison, the theoretical signal-to-noise ratio for a uniform source is also given in the table. This is obtained via equation [1.5] with $c(1) = 1$.

Notice that the number of quantization steps is always an odd number so that the value 0 is always a possible reproduction value[2]. The resultant problem is that, for low values of L, there is a significant difference between $b = \log_2 L$ and the next biggest integer $[\log_2 L]$. Therefore, there is a risk of spilling bits. In the ISO standard,

2. This is for *midtread* quantizers [JAY 84, p. 117].

L	$\mathrm{SNR}_{\mathrm{ISO}}$	$20\log_{10} L$	b
3	7.00	9.54	1.67
5	11.00	13.98	2.33
7	16.00	16.90	3
9	20.84	19.08	3.33
15	25.28	23.52	4
31	31.59	29.82	5
63	37.75	35.99	6
127	43.84	42.08	7
255	49.89	48.13	8
511	55.93	54.17	9
1023	61.96	60.20	10
2047	67.98	66.22	11
4095	74.01	72.25	12
8191	80.03	78.27	13
16383	86.05	84.29	14
32767	92.01	90.31	15
65535	98.01	96.33	16

Table 7.1. *Signal-to-noise ratio as a function of the number of quantization steps*

this problem is solved by grouping three samples if $L \leq 9$. The number of bits used is therefore given by:

$$b = \frac{1}{3}\left[\log_2 L^3\right]$$

The fourth column in Table 7.1 gives the resolution b (the equivalent number of bits per sample) as a function of L.

Table 7.2 indicates the permitted quantizers in each sub-band. They are indicated by the number of quantization steps they have [NOR 93, page 52].

Notice that the sub-bands are split into five groups. The first two groups, corresponding to frequencies between 0 and 7.5 kHz, accept 16 quantizers. It is therefore necessary, in the first instance, to dedicate 4 bits to specify the number of the selected quantizer. This number is denoted as $\mathrm{No}Q_k$. This first two groups are distinct through having a different configuration of possible quantizers. The third group allows 8 quantizers and the fourth allows 4. Sub-bands 27–32 are not coded.

In order to reconstruct the signal, the receiver needs to know not only the code words associated with each component of the vector $\underline{a}_k$ and with the scale factors, but also the bit allocation realized at the transmitter. This last piece of information must therefore be transmitted in the bitstream. It takes $11 \times 4 + 12 \times 3 + 4 \times 2 = 88$ bits. Assuming, for example, that 20 out of the possible 27 sub-bands are transmitted (the

NoQ	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
SB0	0	3	7	15	31	63	127	255	511	1023	2047	4095	8191	16383	32767	65535
SB1	0	3	7	15	31	63	127	255	511	1023	2047	4095	8191	16383	32767	65535
SB2	0	3	7	15	31	63	127	255	511	1023	2047	4095	8191	16383	32767	65535
SB3	0	3	5	7	9	15	31	63	127	255	511	1023	2047	4095	8191	65535
SB4	0	3	5	7	9	15	31	63	127	255	511	1023	2047	4095	8191	65535
SB5	0	3	5	7	9	15	31	63	127	255	511	1023	2047	4095	8191	65535
SB6	0	3	5	7	9	15	31	63	127	255	511	1023	2047	4095	8191	65535
SB7	0	3	5	7	9	15	31	63	127	255	511	1023	2047	4095	8191	65535
SB8	0	3	5	7	9	15	31	63	127	255	511	1023	2047	4095	8191	65535
SB9	0	3	5	7	9	15	31	63	127	255	511	1023	2047	4095	8191	65535
SB10	0	3	5	7	9	15	31	63	127	255	511	1023	2047	4095	8191	65535
SB11	0	3	5	7	9	15	31	65535								
SB12	0	3	5	7	9	15	31	65535								
SB13	0	3	5	7	9	15	31	65535								
SB14	0	3	5	7	9	15	31	65535								
SB15	0	3	5	7	9	15	31	65535								
SB16	0	3	5	7	9	15	31	65535								
SB17	0	3	5	7	9	15	31	65535								
SB18	0	3	5	7	9	15	31	65535								
SB19	0	3	5	7	9	15	31	65535								
SB20	0	3	5	7	9	15	31	65535								
SB21	0	3	5	7	9	15	31	65535								
SB22	0	3	5	7	9	15	31	65535								
SB23	0	3	5	65535												
SB24	0	3	5	65535												
SB25	0	3	5	65535												
SB26	0	3	5	65535												

Table 7.2. *Number of possible quantization steps per sub-band*

last five are never transmitted), the side information (bit allocation + scale factors) represents $88 + 120$ bits. At 96 kbit/s, the number of bits that remain for coding the normalized sub-band signal amplitudes is $96 \times 384/44.1 - 88 - 120 = 628$ bits. At 64 kbit/s, 350 bits are available. We can therefore see that the number of bits reserved for the "noble" part decreases a lot. We can therefore understand why this coder is efficient in reducing the bit rate until reaching a critical bit rate below which the quality is hardly reduced.

7.3. MPEG-2 AAC coder

All of the technical details of this coder can be found in [INT 97]. A more student-friendly description is available in the article [DER 00]. Here we present only the principles.

Figure 7.6 shows a piano signal in the time domain (Figure 7.6(a)) and in the frequency domain (Figure 7.6(b)). This spectral representation is obtained simply by taking the moduli of the components of the vector $\underline{X}(m)$ and expressing them in dB. The transform used in the MPEG-2 AAC coder is the MDCT with $N = 2048$ and $M = 1024$.

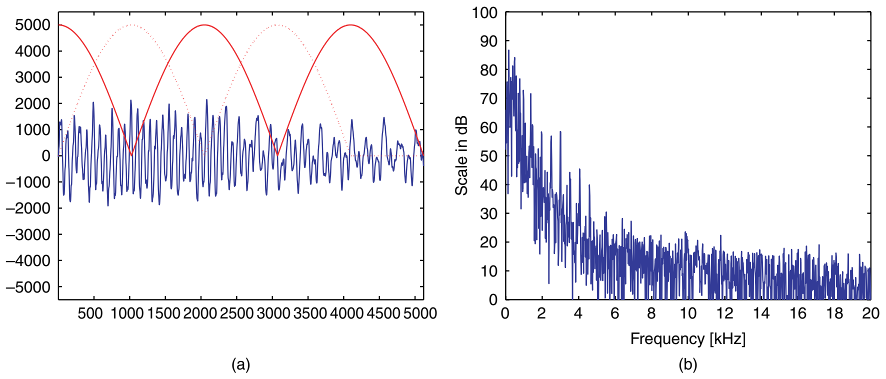

Figure 7.6. *Piano signal (a) in the time domain, (b) in the frequency domain*

The graph in Figure 7.7 still represents $|\underline{X}|$ but on the *x*-axis the frequencies are expressed in Bark and the *y*-axis is linear. This gives the effect of zooming in on the bass frequencies and the high values are strongly amplified.

Firstly, we can observe that the MDCT is a real-valued transform but the values are not necessarily positive. The MPEG-2 AAC coder processes the signs separately [3]. Quantizing the moduli of the components of the vector $\underline{X}(m)$ is carried out as follows. Assume that M parameters $\underline{g} = [g(0) \cdots g(M-1)]$ are known, referred to as the *scale factors*. At the transmitter, a vector of integers is calculated using the formula:

$$\underline{i} = \text{round}\left(\left[\frac{|X(0)|}{g(0)} \cdots \frac{|X(M-1)|}{g(M-1)}\right]\right)$$

Note that at the receiver, knowing $\underline{i}$ and $\underline{g}$, we can reconstruct:

$$[|\hat{X}(0)| \cdots |\hat{X}(M-1)|] = [g(0) \times i(0) \cdots g(M-1) \times i(M-1)]$$

Then we obtain $[\hat{x}(0) \cdots \hat{x}(M-1)]$ by recovering the sign information but a reconstruction error arises from the rounding operations. The graphs in Figure 7.8 show, for a given choice of scale factor, vector $\underline{g}$ (stepped curve on the left-hand graph)

3. In reality, not always! We do not go into the details here. We no longer talk in terms of coding the signs.

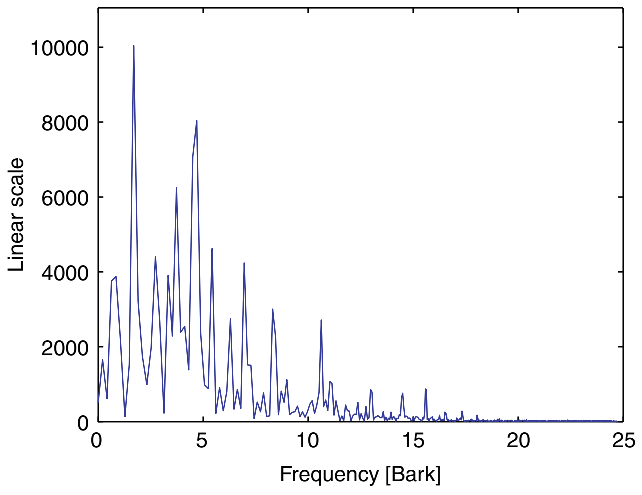

Figure 7.7. *Moduli of components of the vector $\underline{X}(m)$ with frequencies in Bark on the x-axis and a linear y-axis*

and the vector of integers $\underline{i}$ (on the right-hand side). From real values in the range of 0 to 10,000, we pass to integer values between 0 and 4.

Knowing the vector $\underline{i} = [i(0) \cdots i(M-1)]$, we can take this coding problem and ask ourselves what the cost in bits is for this information. In this example, since all the integers are lower than 8, 3 bits are enough to code each integer exactly. The total number of bits which is enough to represent the vector $\underline{i}$ is therefore $M \times 3$. Is

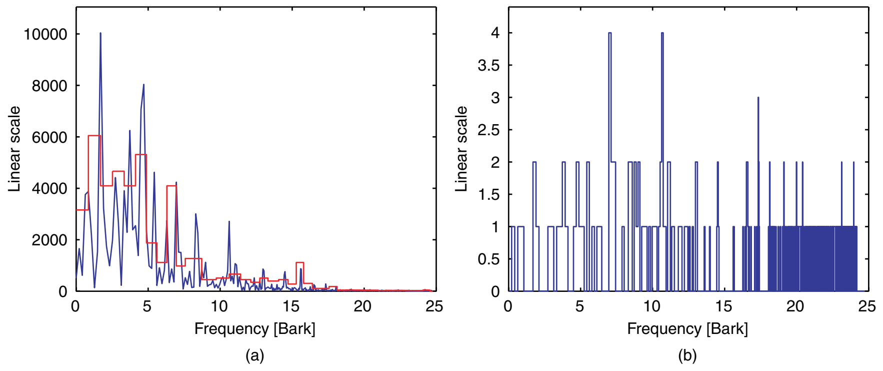

Figure 7.8. *(a) Vectors $|\underline{X}|$ and $\underline{g}$. (b) Vector $\underline{i}$*

this total number of bits necessary? There is certainly a more economical solution using a Huffman coding. In reality, the solution used in this coder is to partition the frequency axis into 51 bands which correspond *grosso modo* to the critical half-bands, to determine the maximum value taken by $i(k)$ in each band and then to realize a separate coding in each band using indexed Huffman tables on the maximum value and constructed value using training data. The bitstream therefore carries code words of the M integers $i(k)$, the code words of the 51 maximum values, and the code words of the scale factors $g(k)$.

Now, only the scale factor vector $\underline{g}$ needs to be determind. *A priori* we can see this problem as a simple optimization problem: it requires determining $\underline{g}$ while minimizing the reconstruction error power under the constraint that the necessary number of bits must be less than the number of bits available. This solution is not in fact realistic since, as we can notice, it requires only low compression rates (of the order of 2) if we want the reconstructed signal to be transparent. To obtain higher compression rates (from 10 to 15), we must make use of the psychoacoustic results and noise shaping techniques. This means, in reality, that the minimization problem has two constraints: one constraint on the bit rate, that the number of bits needed must be less than the number of bits available, and the psychoacoustic constraint $S_Q(f) < \Phi(f)\ \forall f$. The optimization algorithm proposed in the standardization document is an algorithm of the gradient type based on the following property. If the scale factor $g(k)$ is increased, the integer $i(k)$ is reduced. The bit rate is therefore reduced but the reconstruction noise power increases in the frequency zone characterized by the index k. Recall that the proposed algorithm is optional and does not form part of the standard.

As the scale factors must be transmitted, they use up part of the available binary bit rate. It is unreasonable to transmit M scale factors. The previous partition of the frequency axis into 51 bands is also used for scale factors which explains the stepped curve in Figure 7.8. The coding of the first scale factor is realized on 8 bits; the coding of the 50 following factors is realized by coding $\Delta(k) = g(k) - g(k-1)$ using a Huffman table.

Note that the MPEG-2 AAC coder is a closed 5 loop perceptual coder. The algorithm for determining the scale factors makes use of the real value of the reconstruction error and not the assumed value after a rounding operation.

This presentation highlights only the major principles of the method. A more detailed study is necessary if we wish to have a good understanding of this coder. A number of points can be explained further. Let us take just one example and examine the graph in Figure 7.9.

The music signal shown is a piano signal which clearly shows the beat of one note. It shows a significant rise in power. As much as the signal is relatively stationary in an analysis frame, we are without a doubt interested in finding a good frequency

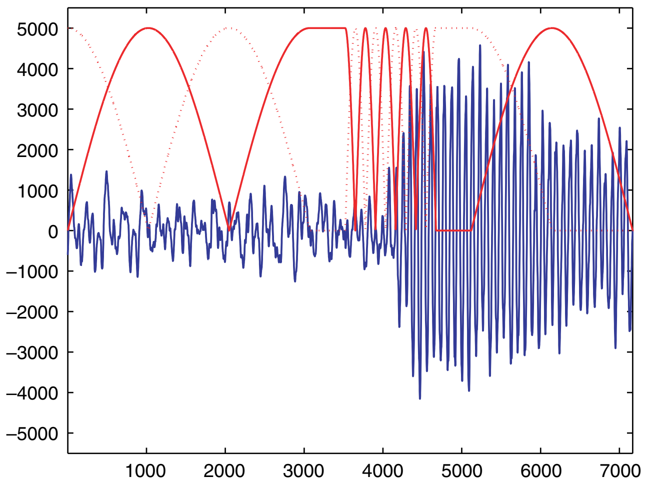

Figure 7.9. *Switching between long and short frames*

resolution which requires a long analysis frame. In the frame(s) where the signal is not stationary, we are rather more interested in finding a good temporal resolution. We therefore need shorter analysis frames. In the MPEG-2 AAC coder, a frame of $N = 2048$ samples is divided into eight sub-frames of 256 samples. In order to respect the perfect reconstruction conditions given by equation [3.2], we have to introduce transition frames upstream and downstream. Note that this requires a criterion which allows switching between long frames and short frames. The proposed criterion[4] in the standardizing document is rather subtle. It makes use of the prediction gain: during a rise in power the signal is not very predictable. This long frame/short frame switching gives good results but has a significant effect on the reconstruction delay. In effect, we can only decide on switching after having completely observed a rise in power and in this case we must introduce a transition frame upstream of the rise in power. Recall that long reconstruction delays are incompatible with bi-directional communications, which explains the existence of a version of the AAC coder called *low delay* in MPEG-4, a version in which the notion of short frames has been dropped.

7.4. Dolby AC-3 coder

This coder [ADV 95] has a classic architecture. The time-frequency transform is realized by an MDCT with overlapping frames of 512 samples. The frames become twice as small if a significant change in power in the second half of the frame

4. NB: this is not part of the standard.

is observed. The novelty in this coder is the use of the floating point format (mantissa + exponent) to code the transform coefficients. A rounding operation is realized on the mantissa with a precision which is a function of the signal-to-mask ratio. The exponent plays the role of the scale factor. The specific bit allocation information is not transmitted directly: it is reconstructed at the decoder from the spectral envelope which is coded like a speech coder.

7.5. Psychoacoustic model: calculating a masking threshold

7.5.1. *Introduction*

In this section, some information is presented as simply as possible to allow us to understand the significance of masking and to calculate a masking threshold. The only psychoacoustic result of interest here is how to deduce from the spectral characteristics of a music signal, the spectral density of a noise which is suitable to be added to the original signal without this noise being perceptible to the listener. For further information on the field of sound perception we can consult, for example, [ZWI 81, MOO 82].

7.5.2. *The ear*

We know that the ear consists of three parts: the outer ear which is formed by the pinna and the auditory canal, the middle ear, comprising the eardrum and the auditory ossicles, and the inner ear which comprises the cochlea. The outer and middle ear carry out a bandpass filtering function for frequencies between 20 Hz and 20 kHz of which the frequency response modulus is directly linked to the absolute threshold of hearing. They also perform an impedance adaptation function between propagation of waves through the medium of air and the medium of liquid in the inner ear. The inner ear consists of the cochlea, which contains a liquid in which the basilar membrane is bathed. This is stimulated by the movement of the stirrup. This membrane can be seen, once unrolled, as a straight segment of a given length (3.5 cm) which has a large number of nerve cells. All sound excitations make this membrane vibrate. A pure sound, a sinusoidal waveform of a given frequency f_1, makes this membrane vibrate around a particular position k_1 of the basilar membrane and this vibration produces excitation in the neurons of the auditory nerve. Thus a frequency to position transformation occurs.

Zwicker [ZWI 81] declares that k_1 is a frequency linked to the frequency f_1 by a non-linear relation. This relation becomes linear if the frequency is expressed on a new scale known as the *Bark scale*. This new frequency scale is directly linked to the notion of critical bands presented in section 7.5.3. If k is the position of a particular fiber along the basilar membrane and $S_E(k)$ is the amount of power measured by this fiber, the *spread function* $S_E(k)$ which is at the beginning of the masking threshold calculation, as we will see in section 7.5.4, reflects the spread and the power of the excitation created in the auditory system by the excitation sound.

Zwicker showed that the hypothesis that he made led to taking a filter bank as a representation of the auditory system (neglecting the initial bandpass filter). In effect, imagine a sinusoid $x(n) = a \sin(2\pi f_1 n)$ entering M filters with frequency responses $H_k(f)$. The M sub-band signals are expressed:

$$y_k(n) = a|H_k(f_1)|\sin(2\pi f_1 n + \phi_k(f_1))$$

They have power $\sigma^2_{Y_k} = \sigma^2_X |H_k(f_1)|^2$. The term $\sigma^2_{Y_k}$ represents the power transmitted by the pure excitation sound of frequency f_1 to the k^{th} position of the basilar membrane. If we plot $|H_k(f_1)|^2$ as a function of k, we obtain the spread function $S_E(k)$ as a function of f_1.

7.5.3. *Critical bands*

Different experiments have led to the same understanding of critical bands and provide approximately the same bandwidths. Only one is described here, which is directly related to the absolute threshold of hearing. Two pure, simultaneous sounds of different frequencies are detected at a power weaker than one of them with the condition that the frequency gap between them is not greater than a given value. More precisely, consider a sinusoidal waveform at a frequency f_1. If the power σ^2_X of this sinusoid satisfies $\sigma^2_X \geq S_a(f_1)$ where $S_a(f)$ is the absolute threshold of hearing, this sinusoidal waveform is audible. Imagine a second sinusoidal waveform with frequency f_2 which is close to f_1 with an identical power to the first one. Experimental measurements show that both of the sinusoidal waveforms are audible provided that $2\sigma^2_X \geq S_a(f_1)$. Consider N sinusoidal waveforms with regularly spaced frequencies $f_1 < f_2 < \cdots < f_N$ and with the same power, σ^2_X. The preceding property – the set of N sinusoidal waveforms is audible provided that $N\sigma^2_X \geq S_a(f_1)$ – is satisfied by the condition that the bandwidth $\Delta f = f_N - f_1$ is less than a threshold known as the critical bandwidth in the neighborhood of the frequency f_1. This experiment shows that, in conclusion, we can accept that the power perceived by the ear in a critical band is the same as the sum of all the powers of the components in this frequency band: the ear integrates powers in a critical band. We can also say that the ear behaves as a filter bank which makes an irregular partition of the frequency axis and then sums the power in each sub-band.

The audible band is divided into 24 critical bands for which the frequency partitioning is given in Table 7.3 [ZWI 81]. We can define a new frequency scale, the Bark scale, [5] which corresponds simply to the critical band number.

5. There is also another scale, the MEL scale, which is approximately 100 times finer than this.

Band number	Lowest frequency	Highest frequency	Width of critical band
1	20	100	80
2	100	200	100
3	200	300	100
4	300	400	100
5	400	510	110
6	510	630	120
7	630	770	140
8	770	920	150
9	920	1080	160
10	1080	1270	190
11	1270	1480	210
12	1480	1720	240
13	1720	2000	280
14	2000	2320	320
15	2320	2700	380
16	2700	3150	450
17	3150	3700	550
18	3700	4400	700
19	4400	5300	900
20	5300	6400	1100
21	6400	7700	1300
22	7700	9500	1800
23	9500	12000	2500
24	12000	15500	3500
25	15500		

Table 7.3. *Critical bands. Frequencies are expressed in Hz*

In this table, the 24 critical bands are put next to one another. In fact, the ear can create a critical band around any frequency. The relation between a frequency expressed in Hz in the range [20 Hz–20 kHz] and a frequency expressed in Bark in the range [1–24] can be approximately expressed as follows:

$$f_{\mathrm{B}} = 13 \times \mathrm{arctg}\left[0.76\frac{f_{\mathrm{H}}}{1000}\right] + 3.5 \times \mathrm{arctg}\left[\left(\frac{f_{\mathrm{H}}}{7500}\right)^2\right]$$

7.5.4. *Masking curves*

The ear has the property of not perceiving a sound in the presence of another sound provided that the characteristics of the two sounds have a particular relation. This is the *masking phenomenon*. There are two types of masking of one sound by another. We speak of temporal masking when two sounds appear consecutively and frequency masking when two sounds appear simultaneously. The temporal masking phenomenon can be of interest in coding but is not presented here. Let us introduce frequency masking.

Recall that a sinusoidal waveform of frequency f_1 is only audible in a perfectly silent environment, under the condition that its power is greater than the absolute threshold of hearing for that frequency. Consider now the case of two sinusoidal waveforms. The first sinusoidal waveform, the masking sinusoidal waveform, has a particular frequency f_1 and a given power $\sigma^2_{X_1}$, much higher than $S_a(f_1)$. For all frequencies f_2 in the audible band, the power $\sigma^2_{X_2}$ of the second sinusoidal waveform, the masked sinusoidal waveform, is measured at the limit of audibility in the presence of the first sinusoidal waveform. The function $\sigma^2_{X_2}(f_2)$ is called the masking curve or the mask effect curve. The form of this masking curve is given in Figure 7.10(a) for a sinusoidal masking sound.

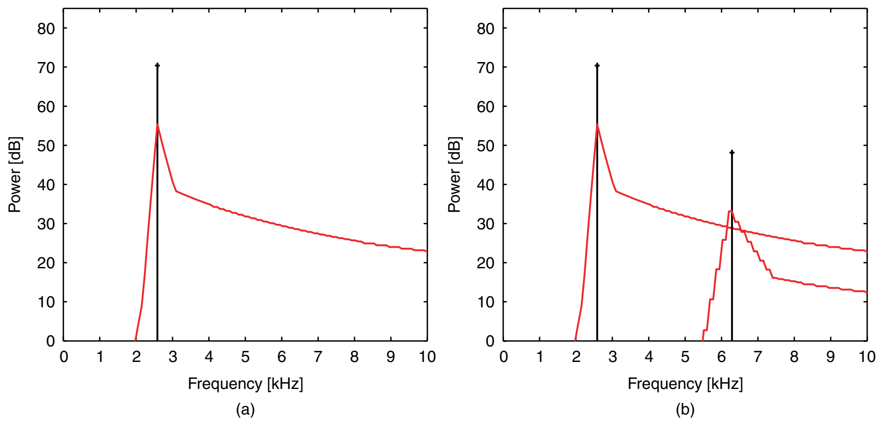

Figure 7.10. *(a) Masking curve when the masking sound is sinusoidal. (b) Influence of frequency on the shape of the masking curve*

The masking curves of one sinusoidal waveform by another sinusoidal waveform are not the only curves of interest. Psychoacoustics also gives us precise results for the case in which the masked signal is a narrow-band noise. Furthermore, a music signal can be considered to be composed of a number of pure sounds, which can be modeled by sinusoidal waveforms, and of impure sounds, which can be modeled by narrow-band noise, and it is necessary to examine the following four cases: a sinusoid masked by a sinusoid, a sinusoid masked by a narrow-band noise, and more particularly a narrow-band noise masked by a sinusoid and a narrow-band noise masked by a narrow-band noise. The curves shown in Figure 7.10 are, in reality, the masking curves of a sinusoidal waveform by a narrow-band noise and have been obtained via simulation by applying the number 1 psychoacoustic model from MPEG.

All of these curves, whatever case is considered[6] have the same appearance in a triangular shape. Let us examine these curves more exactly and study the influence of the parameters f_1 and $\sigma^2_{X_1}$ on $\sigma^2_{X_2}(f_2)$. The first property of these curves is that they reach a maximum at the frequency f_1. We notice in Figure 7.10 that the power $\sigma^2_{X_2}$ at frequency f_1 is lower than the power $\sigma^2_{X_1}$. The difference $\sigma^2_{X_2}(f_2 = f_1) - \sigma^2_{X_1}$ is called the *masking index*. The second property is that they show a significant assymmetry. The gradient of the curves is steeper toward frequencies lower than the frequency of the masking sound than toward the higher frequencies. The gradient of the curves depends on the frequency f_1 of the masking sound. It becomes less steep as the frequency f_1 increases. If the frequency is expressed in Bark and the power in dB, we can show that the masking curves can be modeled as straight lines which have gradients that are independent of the frequency f_1. Unfortunately, the gradient at higher frequencies remains dependent on $\sigma^2_{X_1}$. It is, however, less steep than $\sigma^2_{X_1}$, which is large.

7.5.5. *Masking threshold*

The preceding exposition is not yet enough for us since it is applied only to the case when the masking and masked sounds are pure sounds or narrow-band noises with bandwidth less than or equal to that of a critical band. On the one hand, an audio signal is composed of an infinity of contributions: the power spectral density $S_X(f)$ characterizes the relative importance of each contribution. On the other hand, we are looking to determine a signal at the limit of audibility, the quantization noise, and this signal also has an infinity of contributions.

Two types of problems arise. We are in the presence of numerous masking sounds: how can we "add them up" to determine the power of the masked sound right at the limit of audibility? We are in the presence of numerous masked sounds: how can we generalize them? These are difficult problems; they are addressed briefly in the literature written by experts in acoustics since they are specifics of coding application[7]. Psychoacoustic experts do not know how to describe the masking phenomenon with precision when the number of masking components is greater than 2 or 3 while we need to be able to process the case of 512 or even 1024 masking components. This is the problem of the "laws of additions" in psychoacoustics as we have just mentioned. The available coding models have been determined by engineers

6. Except for masking a sinusoid by a sinusoid, since it introduces beating phenomena when two frequencies of pure sounds are close, but is useless in coding.

7. And since recently in watermarking applications.

who have defined the numerical value of a few parameters after a very large number of listenings with a complete coder.

An example of the masking threshold in the band [0–10] kHz calculated from a violin signal is given in Figure 7.3. It uses the number 1 psychoacoustic model of the MPEG-1 standard provided as an option in the standardization document.

Chapter 8

Audio Coding: Additional Information

The development of high-quality, reduced bit rate perceptual coders has been the subject of strenuous research efforts for the past 20 years. At the time of writing, the most successful coder is the Audio Advanced Coder (AAC) in MPEG-2 and MPEG-4. Improvements, primarily through the introduction of new tools, are still being made. In this chapter, two recent and characteristic developments are presented: research into good-quality, low bit rate coders and very good quality or even lossless coders. Throughout, we note that these developments do not at all deliver the same compression rates and are consequently not of the same level of interest. Making available a low bit rate coder while maintaining an acceptable quality is of considerably greater interest to operators than making available of lossless coder that has a low compression rate. That being said, the lossless coder is presented in this book because it has an interesting property from an educational perspective: the algorithms used in this particular coder are directly adapted from algorithms developed in image coding. This convergence of audio coding and image coding is sufficiently rare to be worth noting.

In this chapter, these recent developments of the AAC are presented relatively briefly. In the following chapter, a more extensive study of the dedicated stereo coding tool is presented.

8.1. Low bit rate/acceptable quality coders

The audio coder HE-AACv2 developed in the MPEG-4 framework and the ESTI 3rd Generation Partnership Project (3GPP) is built around the MPEG-4 AAC and makes use of the two tools called *Spectral Band Replication* (SBR) and *parametric* *stereo* (PS). It aims for bit rates in the range 8-kb/s mono to 48-kb/s stereo (or in 5.1 format or *surround* at 64 kb/s).

8.1.1. *Tool one: SBR*

This band extending technique was proposed in 1999 in the standardization framework of AM band digital radio (*Digital Radio Mondial* (DRM) consortium). It was standardized in 2003. It involves the reconstruction of the high signal frequencies at each instant from the low-frequency content of the signal. For example, the signal of a maintained periodic sound consists of a fundamental frequency and a series of harmonics which have frequencies that are multiples of the fundamental frequency as shown in Figure 8.1. The position of the harmonics along the frequency axis does not need to be transmitted. Only the amplitudes of the harmonics need to be coded. In general, this is realized by coding the calculated spectral envelope, for example, using LPC analysis. We have seen that coding the spectral envelope, for a speech coder, requires approximately 1 kb/s. The majority of the available bit rate is therefore reserved for coding bass frequency content, while the high-frequency part generally requires only a few kilobits per second.

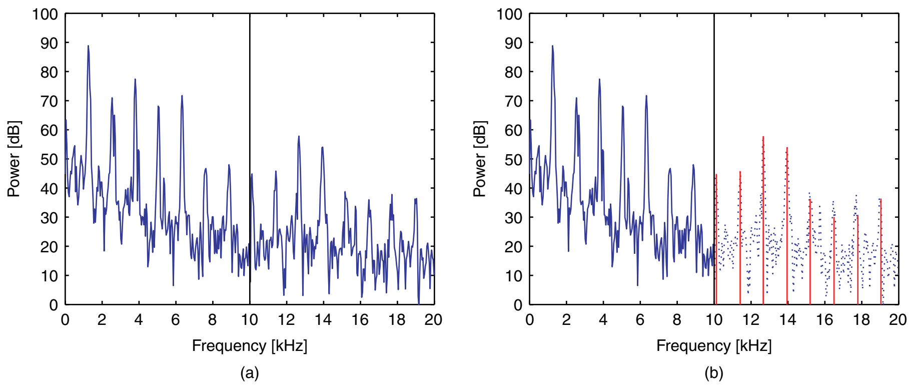

Figure 8.1. *Possibility of reconstructing the high-frequency spectrum from bass frequencies*

This method's principle is as follows:

– The input signal $x(n)$ is downsampled, for example, by a factor of 2, and then coded by a standard coder, which provides the first part of the bitstream. If this coder produces a reasonable quality at a bit rate of D bit/s when applied to full-band signals, we can reasonably assume that it will need $D/2$ bit/s when it uses a half-band signal. In the framework of the HE-AACv2 audio coder, this is true for the AAC. The AAC in operation is almost the normal AAC except for the fact that the psychoacoustic model is modified according to the requirements of this coder.

– The input signal is also transformed into the frequency domain and analyzed. In the framework of the HE-AACv2 audio coder, the input signal is processed by a bank of 64 QMF filters which provide the sub-band signals. The second part of the bitstream codes the characteristic information for the spectral envelope, the presence or absence of harmonics, and the power ratio between tonal and non-tonal components.

8.1.2. *Tool two: PS*

8.1.2.1. *Historical overview*

Ever since the audio compression problem arose, that is in the middle of the 1980s, several techniques have been developed to code signals in stereo. The initial step was to realize a perceptual coding on each of the two channels, but this separate coding required prohibitive bit rates, and two techniques were quickly established:

– *M/S stereo coding*: instead of transmitting the left and right channels, the normalized sum (M: *middle channel*) and the difference (S: *side channel*) are transmitted. Coding is realized selectively in the frequency domain. This process is reversible (which is an advantage), but the compression rate is a function of the signal (which is a disadvantage). The compression rate is at its maximum of 2 when the two channels, left and right, are practically equal, but this is not the most frequent case. For multi-channel audio signals, this process can be used by a pair of speakers.

– *Intensity stereo coding*: above 2 kHz, the human auditory system is not sensitive to phase differences between the left and right signals of a stereo recording. The *intensity stereo* technique makes use of this property by only transmitting in the high frequencies a single signal and a few parameters. These parameters, function of time and frequency, code the intensity difference between the two stereo channels. This makes use of the fact that the perception of high-frequency components is directly linked to the temporal energy envelope. Considering that the coding is carried out in the high-frequency half-band, the gain and bit rate are of the order of 25% for stereo signals. The gain may be higher for multi-channel signals because coding is carried out only in a single channel (in the high-frequency half-band).

A renewed interest in multi-channel audio compression (stereo, and also for the 5.1 format) has been evident for the past few years, driven by the DVD market and telephone operators who want, for example, good-quality sound reproduction systems on mobile telephones.

8.1.2.2. *Principle of PS audio coding*

Figure 8.2 shows the principle of the parametric audio coder.

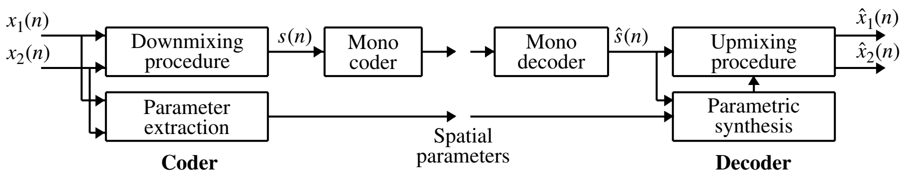

Figure 8.2. *Parametric audio coder operating principle*

When the target bit rate is low (a few dozen kilobits per second), only one possibility really exists: constructing a single signal from all the channels (*downmixing* procedure), coding the signal with a standard coder, for example, the MPEG-4 AAC, and transmitting this information with side information. The following section focuses on the spatial perception of sound and shows that in reality, there are three important types of side information: the inter-channel time difference (ICTD), the inter-channel level difference (ICLD), and the inter-channel coherence (ICC). This step can be interpreted as a generalization of the *Intensity Stereo* technique. At the receiver, a spatial image is reconstructed by making a stereo or multi-channel signal (*upmixing* procedure) from the coded mono signal and the side information. As the side information is independent of the coder, this technique can be applied generally. It is also of great interest for ensuring complete backward compatibility with a mono transmission. We can also note that masking conditions are not necessarily of interest in a binaural context.

Two particular and separate teams have worked in this field and proposed two similar techniques. Faller *et al.* [BAU 03, FAL 03, FAL 04] named their proposal "binaural cue coding". Breebart *et al.* [BRE 05] called their technique "parametric stereo". Teams have also put forward their contributions, for example [BRI 07, LAP 06]. References to MPEG works can be found in [HER 04, PUR 04].

8.1.2.3. *Results*

The parametric description of the spatial sound field enables us to obtain an extremely compact representation (a few kilobits per second). This technique was developed in the MPEG-4 framework, especially for the HE-AACv2 coder to obtain a total bit rate of 24 kb/s [PUR 04]. This procedure therefore enables an extremely high compression rate. Listening tests [PUR 04] show that this 24-kb/s coder is approximately equivalent to the HE-AAC stereo coder at 32 kb/s, which is itself approximately equivalent to the 64-kb/s AAC stereo coder. The quality is not perfect but acceptable. This is called "intermediate" quality.

Recently, this technique was generalized in the 5.1 format. This is the MPEG Surround coder. We can say that the bit rate required for the 5-channel format was in the order of 300 kb/s in 1994 with the MPEG-2 layer 3 coder, 200 kb/s in 2000 with the MPEG-4 AAC, 128 kb/s in 2004 with the HE-AAC, and 64 kb/s today with this new parametric audio coding technology.

8.1.3. *Sound space perception*

We just provide a few psychoacoustic results relating to binaural hearing in the simplest case, a sound source in an open field [CAL 02].

The incident direction of the sound is characterized by its azimuth (angle between the incident direction and the vertical half-plane located in the axis of the face) and its elevation (angle between the incident direction and the horizontal plane). The elevation positioning, which gives very different results on the nature of the signal and its positioning in terms of distance which is linked to the perceived intensity (for a pure sound, the subjective distance is rather more important than a higher intensity), is not involved in the following description.

Two binaural indices are used in azimuthal positioning:

– Interaural time difference (ITD): the difference in the sound's arrival time at the left ear and the right ear. This is particularly important for bass frequencies, frequencies less than 1,500 Hz (the wavelength is 23 cm, which corresponds to the interaural distance).

– Interaural level difference (ILD): the attenuating effect of the head gives rise to a difference in intensity. A sound source positioned to the listener's left will have a greater signal intensity at the left ear than the right ear. This is important particularly at high frequencies.

The third binaural index used in coding is related to the subjective spread of a source: an auditory event may be perceived crisply, or by contrast, it may not be very localized, for example, in a concert in a church as a result of multiple reflections. This is known as *interaural coherence* (IC).

There are other psychoacoustic elements relevant to the spatial perception of sound, but they are hardly used in audio coding. Here are a few examples:

– precedence effect or the law of the first wavefront: two sounds arriving within a short time of one another at the ears are heard as a single sound, and the first one determines the spatial perception;

– if the delay is more than a few milliseconds, the reflections of a sound source from obstacles in the vicinity (when they can be separately perceived) are called echoes while subsequent reflections, which are usually fused together, form the reverberation;

– the audio signal entering the auditory system can be regarded as a filtered version of the signal coming from the source. This filtering is characterized by a transfer function (*head-related transfer functions*, HRTF) which is dependent on the source. The head and pinna together create a complex filtering with a noticeable effect between 500 and 16,000 Hz.

The psychoacoustic results therefore provide indices relevant to the signals as they enter the auditory system. When listening to headphones, the indices measured directly from the audio signals are practically identical to the binaural indices. They are no longer even *a priori* needed for a retransmission via loud-speakers because the acoustic transfer functions of the transducers/ears (HRTF) must be considered. In reality, in a compression operation, the aim is not to closely interpret the role of these indices in constructing the spatial effects but to make sure that the indices that are relevant to the signal reconstructed at the receiver are approximately the same as those of the original signal. Since similar indices to the ITD, ILD, and IC are calculated directly from the signal source, new nomenclature is preferred. The new names are ICTD, ICLD, and ICC. For a compression application, what matters is coding and correctly reconstructing the three inter-channel parameters that are similar to the binaural parameters of the original and reconstructed signals.

8.2. High bit rate lossless or almost lossless coders

8.2.1. *Introduction*

A lossless coding never involves compression rates higher than 2 or 3. Studies show that a medium compression rate on an important corpus is of the order of 2 for signals sampled at 48 kHz and quantized on 16 bits, slightly higher for signals sampled at 96 kHz, slightly less for signals quantized on 24 bits. This type of coding is especially necessary for professional (post-production) applications where an AAC music file cannot be decoded, to process it and then to save it in AAC format and start over. Several successive encoding/decoding operations (*tandem coding*) degrade the signal quite quickly (after 2 or 3 tandems). If an improvement layer of 384 kb/s is added to the AAC layer (at 128 kb/s in stereo), no noticeable degradation is observed for a large number of encoding/decoding stages. The main applications are therefore in studios and in archiving.

Until recently [HAN 01], the lossless compression systems available (AudioPak, LTAC, Musiccompress, and so on) most often used a predictive model in the form:

$$y(n) = x(n) - Q\left[\sum_{k=1}^{P} \hat{b}_k x(n-k) - \sum_{k=1}^{Q} \hat{a}_k y(n-k)\right]$$

where the essential quantizing operation $Q(.)$ enables $y(n)$ to be kept in the same format as $x(n)$ and the calculation of the coefficients a_k and b_k is realized using classic algorithms, for example, the Levinson algorithm calculated with fixed point arithmetic. These compression systems then realize an entropy coding of the prediction error via Huffman coding but more often by Rice coding which is adapted for a Laplacian source, that is, a source where the lowest values are the most frequent.

Having seen the importance of this type of coding in these coders, we present the principle of Rice coding by using an example. Let us consider the example of coding the value +26, which is represented by $0 \cdots 011010$ in binary: the ellipsis conveys the fact that supplementary information is required to know more precisely the maximum permitted value. Let us choose an integer m, for example, 3. The code word is then constructed as follows: first, the sign is coded, here with the bit 0, and then the 3 least significant bits, that is, 010, are used directly. Next, we represent the most significant 11 bits. Translating from binary to decimal we obtain 3, and we therefore add 3 zeros followed by a final 1 which gives, at least, the code word 001000001. Knowledge of the maximum possible value is not needed here. Evidently, lower values must have higher probabilities of appearing for this method to be of interest. The parameter m is constant in an analysis frame and can be taken as being equal to $m = \log_2(\log_e(2)\mathbb{E}\{Y(n)\})$. The Rice coding is a special case of Golomb coding when the parameter m is a power of 2, which makes its implementation much easier.

8.2.2. *ISO/IEC MPEG-4 standardization*

8.2.2.1. *Principle*

A lossless or scalable coder with a reasonably fine granularity between lossless and the AAC coding at a bit rate of approximately 64 kb/s per channel was recently standardized [GEI 06, YU 06]. The principle of this process is as follows. The input signal is first coded using the AAC (*core layer*), the difference between the input signal and the reconstructed signal is coded next in a hierarchical fashion until lossless coding is achieved, and this produces the *enhancement layer*. The two layers are then multiplexed.

With regard to the enhancement layer, a time/frequency transform is performed on the input signal by an IntMDCT which consists of a reversible MDCT integer approximation. The subtraction of the result by the AAC is realized in the frequency domain. Next, the error from the two codings is coded by bit plane: a Golomb coding and an arithmetic coding “context coded”. A psychoacoustic model is used from the most significant bit plane toward the least significant bit plane: the reconstruction error must keep a spectral shape approximately equivalent to the masking threshold when the bit rate is increased and when the coding becomes lossless.

8.2.2.2. *Some details*

First, note that the MDCT transform can be broken into a set of elementary rotations, each rotation characterized by a rotation matrix which can itself be broken down as follows:

$$\begin{bmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{bmatrix} = \begin{bmatrix} 1 & \alpha \\ 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 \\ \beta & 1 \end{bmatrix} \begin{bmatrix} 1 & \alpha \\ 0 & 1 \end{bmatrix}$$

with $\alpha = (\cos\theta - 1)/\sin\theta$ and $\beta = \sin\theta$. The transform:

$$\begin{bmatrix} X \\ Y \end{bmatrix} = \begin{bmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix}$$

is calculated in three consecutive stages:

$$z = x + [\alpha y], \quad Y = y + [\beta z], \quad X = z + [\alpha Y]$$

which can be directly inverted:

$$z = X - [\alpha Y], \quad y = Y - [\beta z], \quad x = z - [\alpha y]$$

If at each stage we include a rounding operation, we obtain a transform which, starting from integer values, gives integer values and which is invertible:

$$\hat{z} = x + Q[\alpha y], \quad \hat{Y} = y + Q[\beta \hat{z}], \quad \hat{X} = \hat{z} + Q[\alpha \hat{Y}]$$

By using this property based on the lifting scheme proposed by Daubechies and Sweldens in 1996 or on ladder networks (Bruekers and Enden, 1992), we obtain the IntMDCT. The difference between the power spectra of music signals calculated from the MDCT and the IntMDCT shows that the IntMDCT is a good approximation: the difference is white, approximately 60 dB lower in power than the original signal. Some recent results show that the error can be made more powerful in the lower frequencies than in the higher frequencies.

With regard to the subtraction in the frequency domain, calculating the residual error between the values $c(k)$ of the spectrum calculated from the IntMDCT and their counterparts $\hat{c}(k)$ from the AAC requires that precautions are taken as the invertibility of the AAC must be ensured. The values taken by $\hat{c}(k)$ must be integer, and if the integer associated with the code word specified by the AAC by the index k is denoted $i(k)$, the ratio between $i(k)$ and $\hat{c}(k)$ must be deterministic (using interpolation tables).

Entropy coding makes use of Golomb coding (adapted to Laplacian sources) by bit plane (from most significant bit (MSB) to least significant bit (LSB) to create an overlapping bitstream). The use of arithmetic coding is based on the context (splitting the spectral data into 3 bands (<4 kHz <11 kHz).

Chapter 9

Stereo Coding: A Synthetic Presentation

In the previous chapter, the principle of parametric stereo coding alone was presented. In this chapter, a more precise study of this type of coding is made. The article by Breebaart [BRE 05] is an excellent introduction, but reading this article is not always particularly easy since certain important results are often presented without the luxury of details and since these results are sometimes only very good approximations for an expert, but far from clear to a beginner.

9.1. Basic hypothesis and notation

The two stereo channels are denoted as $x_1(n)$ and $x_2(n)$. To be able to link these results with those of the standard signal processing, it is useful to be able to interpret these two signals as though they were realizations of zero mean stationary ergodic random processes $X_1(n)$ and $X_2(n)$ with powers $\mathbb{E}\{X_1^2(n)\} = \sigma_{X_1}^2$ and $\mathbb{E}\{X_2^2(n)\} = \sigma_{X_2}^2$ and normalized intercovariance function:

$$\rho(n_0) = \frac{\mathbb{E}\{X_1(n)X_2(n+n_0)\}}{\sqrt{\sigma_{X_1}^2\sigma_{X_2}^2}}$$

We can note that:

$$\Gamma = \mathbb{E}\left\{\begin{bmatrix} X_1(n) \\ X_2(n) \end{bmatrix} [X_1(n)X_2(n)]\right\} = \begin{bmatrix} \sigma_{X_1}^2 & \rho(0)\sigma_{X_1}\sigma_{X_2} \\ \rho(0)\sigma_{X_1}\sigma_{X_2} & \sigma_{X_2}^2 \end{bmatrix}$$

$$\Gamma = \sigma_{X_1}\ \sigma_{X_2}\Gamma' = \sigma_{X_1}\sigma_{X_2}\begin{bmatrix} c & \rho(0) \\ \rho(0) & 1/c \end{bmatrix} \qquad [9.1]$$

the intercovariance matrix by setting $c = \sigma_{X_1}/\sigma_{X_2}$.

The three inter-channel indices presented in the preceding chapter, the level difference ICLD, the time difference ICTD, and the coherence ICC are obtained from an estimation of the powers and the intercovariance function:

$$\begin{aligned} \text{ICLD} &= 10\log_{10}\frac{\hat{\sigma}^2_{X_1}}{\hat{\sigma}^2_{X_2}} = 20\log_{10} c \\ \text{ICTD} &= \arg\max_{n_0} |\hat{\rho}(n_0)| \\ \text{ICC} &= |\hat{\rho}(n_0 = \text{ICTD})| \end{aligned} \qquad [9.2]$$

The ratio of the powers is expressed in dB. It usually takes its values from the range [–40, +40] dB. The ICTD is expressed as a number of samples. The ICC, between 0 and 1, gives the extent of the auditory input. Near to 1, the auditory input is perceived to be relatively compact. Close to 0, two distinct inputs are perceived. At approximately 0.4, the result is a balance sensation of the extent to give an order of size.

In reality, psychoacoustic studies show that the calculation of these inter-channel indices must be realized in the time/frequency plane with temporal updating every 30 to 50 ms and a frequency resolution which is homogeneous to the Bark scale. This is the most fundamental hypothesis of this problem. The time/frequency decomposition which is the best adapted therefore seems to be that obtained by a bank of cochlear filters. We can note that this decomposition must be realized at the transmitter as well as at the receiver. Since use of a bank of cochlear filters cannot be envisaged for reasons of processing complexity and also because of the difficulty of defining the bank of synthesis filters which will assure perfect reconstruction, we must therefore go back to using a bank of filters or a simpler transform, for example the discrete Fourier transform.

To avoid complicating the notation, we write the vector constructed from the signal $x(n)$ observed in the current analysis frame of N samples as $\underline{x} = [x(0) \cdots x(N-1)]^t$ and the transformed vector [1] as $\underline{X} = A\underline{x}$. The transformation which is usually used is the short-term Fourier transform with an overlapping weighting window. To simplify the formal description, we assume that A is the Fourier matrix [2] of $N \times N$ dimensions which satisfies $A^t A^* = I$.

1. Conflicting notation: $X(n)$ specifies a random process and $\underline{X}$ a transformed vector. Since references to the random process are infrequent, this conflict is not very problematic.

2. The definition of the DFT involves a non-normalized matrix. Here, we assume, for purposes of simplicity, that it is normalized.

9.2. Determining the inter-channel indices

9.2.1. *Estimating the power and the intercovariance*

Standard estimators of the signal power and the intercovariance between two signals are used. For the power, we find:

$$\hat{\sigma}_X^2 = \frac{1}{N}\sum_{n=0}^{N-1} x^2(n) = \frac{1}{N}\underline{x}^t\underline{x} = \frac{1}{N}\underline{X}^t\underline{X}^* = \frac{||\underline{X}||^2}{N}$$

and for the coefficient $\hat{\rho}(0)$:

$$\hat{\rho}(0) = \frac{\sum_{n=0}^{N-1} x_1(n)x_2(n)}{\sqrt{\sum_{n=0}^{N-1} x_1^2(n)\sum_{n=0}^{N-1} x_2^2(n)}} = \frac{\underline{X}_1^t\underline{X}_2^*}{||\underline{X}_1||\,||\underline{X}_2||}$$

In reality, we want to estimate the intercovariance function for $n_0 \neq 0$. It seems convenient to introduce the vector $\underline{x}_3$ deduced from $\underline{x}_2$ by a circular shift (to the left):

$$\underline{x}_3(n_0) = [x_2(n_0)\cdots x_2(N-1)x_2(0)\cdots x_2(n_0-1)]^t$$

We find:

$$\hat{\rho}(n_0) = \frac{\underline{x}_1^t\,\underline{x}_3(n_0)}{||\underline{x}_1||\,||\underline{x}_3(n_0)||} = \frac{\underline{x}_1^t\,\underline{x}_3(n_0)}{||\underline{x}_1||\,||\underline{x}_2||} = \frac{\underline{X}_1^t\,\underline{X}_3^*(n_0)}{||\underline{X}_1||\,||\underline{X}_2||}$$

Since:

$$X_3(k,n_0) = \sum_{n=0}^{N-1} x_2(n+n_0 \bmod N)\exp\left(-j2\pi\frac{k}{N}n\right) = X_2(k)\exp\left(j2\pi\frac{k}{N}n_0\right)$$

we have:

$$\underline{X}_3^*(n_0) = \text{diag}\left[1\ \exp\left(-j2\pi\frac{1}{N}n_0\right)\cdots\exp\left(-j2\pi\frac{N-1}{N}n_0\right)\right]\ \underline{X}_2^* \quad [9.3]$$

Let us give the expression for the scalar product $\underline{X}_1^t\underline{X}_2^*$ by using the Hermitian symmetry property. We have:

$$\underline{X}_1^t\underline{X}_2^* = X_1(0)X_2(0) + X_1\left(\frac{N}{2}\right)X_2\left(\frac{N}{2}\right) + \sum_{k=1}^{N/2-1}[X_1(k)X_2^*(k) + X_1(N-k)X_2^*(N-k)]$$

$$\underline{X}_1^t \underline{X}_2^* = X_1(0)X_2(0) + X_1\left(\frac{N}{2}\right) X_2\left(\frac{N}{2}\right) + \sum_{k=1}^{N/2-1} [X_1(k)X_2^*(k) + X_1^*(k)X_2(k)]$$

$$\underline{X}_1^t \underline{X}_2^* = X_1(0)X_2(0) + X_1\left(\frac{N}{2}\right) X_2\left(\frac{N}{2}\right) + 2\Re \sum_{k=1}^{N/2-1} X_1(k)X_2^*(k)\}$$

The stereophonic signal is shown in Figure 9.1(a) and the corresponding intercovariance function in Figure 9.1(b). For this stereophonic signal, the inter-channel level difference is approximately 0 dB, the time difference is eight samples, corresponding to 0.2 ms, and the coherence is slightly higher at 0.8, which are characteristic values for a localized sound source approximately equidistant from the two loudspeakers.

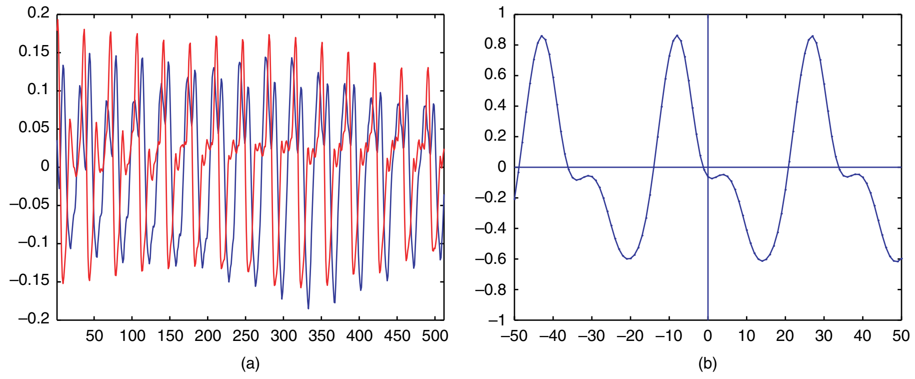

Figure 9.1. *(a) Stereo signal and (b) corresponding intercovariance function*

9.2.2. *Calculating the inter-channel indices*

The inter-channel indices are calculated in each "tile" of the time/frequency decomposition. We write $\underline{X}(b)$ for the sub-vector of $\underline{X}$ specified in the critical band b:

$$\underline{X}(b) = [X(k_b) \cdots X(k_{b+1} - 1)]^t$$

where k_b specifies the first component of the b^{th} critical band. It only matters that the corresponding components of the frequencies are between 0 and $f_e/2$ but we must not forget to take into account, when necessary, the components between $f_e/2$ and f_e. Hence, a scalar product in the form $\underline{X}_1^t \underline{X}_2^*$ which provides a real value must be written:

$$\underline{X}_1^t(b)\underline{X}_2^*(b) + \underline{X}_2^t(b)\underline{X}_1^*(b) = 2\Re\{\underline{X}_1^t(b)\underline{X}_2^*(b)\}$$

in the sub-band domain to continue producing a real value.

We find:

– the level difference:

$$\mathrm{ICLD}(b) = 10\log_{10}\frac{\hat{\sigma}^2_{X_1}}{\hat{\sigma}^2_{X_2}} = 10\log_{10}\frac{\sum_{k=k_b}^{k_{b+1}-1}|X_1(k)|^2}{\sum_{k=k_b}^{k_{b+1}-1}|X_2(k)|^2} = 10\log_{10}\frac{||\underline{X}_1(b)||^2}{||\underline{X}_2(b)||^2}$$

– the time difference:

$$\mathrm{ICTD}(b) = \arg\max_{n_0\in\{-n_0^{\max}\cdots n_0^{\max}\}}|\hat{\rho}(n_0)|$$

with:

$$\hat{\rho}(n_0) = \frac{\Re\{\underline{X}_1^t(b)\ \underline{X}_3^*(b,n_0)\}}{||\underline{X}_1(b)||\ ||\underline{X}_2(b)||}$$

and $n_0^{\max}$ the smallest integer such that:

$$\frac{k_b}{N}n_0^{\max} \geq 1$$

– and the coherence:

$$\mathrm{ICC}(b) = |\hat{\rho}(n_0 = \mathrm{ICTD}(b))|$$

To avoid both a maximization which is costly in terms of calculations and the processing of delicate precision problems[3], the ICTD calculation can be replaced with that for the inter-channel phase difference.

$$\mathrm{ICPD}(b) = \arg\{\underline{X}_1^t(b)\ \underline{X}_2^*(b)\}$$

based on the following justification. The use of the coherence is important if it has a value close to 1. This is the case when a signal is a shifted version of another signal. If we write the Fourier transform of a signal as $X(f)$, the Fourier transform of the signal $x(n+n_0)$ is equal to $X(f)\exp(j2\pi f n_0)$. In this case, the ICTD and the ICPD are directly related since:

$$\arg\{\underline{X}^t\ \underline{X}^*\exp(-j2\pi f n_0)\} = -2\pi f n_0$$

which leads to:

$$\mathrm{ICPD} = -2\pi f\ \mathrm{ICTD}$$

3. For critical bands corresponding to high frequencies $n_0^{\max}$ is low. Fractional values of n_0 are therefore of interest to obtain good precision.

Knowing the ICPD, we can deduce the ICC. In effect, since:

$$\underline{X}_3^*(b) = \text{diag}\left[\exp\left(-j2\pi\frac{k_b}{N}n_0\right)\cdots\exp\left(-j2\pi\frac{k_{b+1}-1}{N}n_0\right)\right]\ \underline{X}_2^*(b)$$

conforms with equation [9.3], we obtain the approximation:

$$\underline{X}_3^*(b) = \exp[-j\ \text{ICPD}(b)]\ \underline{X}_2^*(b)$$

We therefore have:

$$\underline{X}_1^t(b)\underline{X}_3^*(b) = \underline{X}_1^t(b)\underline{X}_2^*(b)\exp[-j\ \text{ICPD}(b)]$$

$$\underline{X}_1^t(b)\underline{X}_3^*(b) = |\underline{X}_1^t(b)\underline{X}_2^*(b)|\exp[j\ \text{ICPD}(b)]\exp[-j\ \text{ICPD}(b)] = |\underline{X}_1^t(b)\underline{X}_2^*(b)|$$

which leads to:

$$\text{ICC}(b) = \frac{|\underline{X}_1^t(b)\ \underline{X}_2^*(b)|}{||\underline{X}_1(b)||\ ||\underline{X}_2(b)||}$$

9.2.3. *Conclusion*

Finally, for each time/frequency decomposition tile, knowing $\underline{X}_1(b)$ and $\underline{X}_2(b)$, the three inter-channel indices can be calculated:

$$\text{ICLD}(b) = 10\log_{10}\frac{||\underline{X}_1(b)||^2}{||\underline{X}_2(b)||^2}$$

$$\text{ICPD}(b) = \arg\{\underline{X}_1^t(b)\ \underline{X}_2^*(b)\}$$

$$\text{ICC}(b) = \frac{|\underline{X}_1^t(b)\ \underline{X}_2^*(b)|}{||\underline{X}_1(b)||\ ||\underline{X}_2(b)||} \qquad [9.4]$$

9.3. Downmixing procedure

At the transmitter, two stereo channels $x_1(n)$ and $x_2(n)$ are available. We are looking for the monophonic signal which can be coded as follows. The solution which immediately comes to mind is to construct the signal $s(n) = [x_1(n) + x_2(n)]/\sqrt{2}$. This solution[4] is not really the best since it does not even guarantee the preservation of power when $\rho(0) \neq 0$:

$$\sigma_S^2 = \frac{1}{2}\mathbb{E}\ \{[X_1(n) + X_2(n)]^2\ \} = \frac{1}{2}(\sigma_{X_1}^2 + \sigma_{X_2}^2 + 2\rho(0)\sigma_{X_1}\sigma_{X_2})$$

4. Called passive *downmix* in the literature.

This expression can even be zero if $\rho(0) = -1$. The general idea is to determine $s(n)$ via an intermediate rotation written:

$$\begin{bmatrix} s(n) \\ s'(n) \end{bmatrix} = \begin{bmatrix} \cos(\theta) & \sin(\theta) \\ -\sin(\theta) & \cos(\theta) \end{bmatrix} \begin{bmatrix} x_1(n) \\ x_2(n) \end{bmatrix} \qquad [9.5]$$

and by determining the angle θ which maximizes the power of $s(n)$. In reality, the process that is presented to begin with in the time domain is realized in the frequency domain, more precisely for each tile in the time/frequency decomposition. An overlap-add synthesis brings the signal $s(n)$ back into the time domain.

9.3.1. *Development in the time domain*

Let us determine the (normalized) vector $\underline{p} = [\cos(\theta)\ \ \sin(\theta)]^t$, which maximizes the power of $S(n) = \underline{p}^t[X_1(n)X_2(n)]^t = \underline{p}^t\underline{X}(n)$. We find:

$$\underline{p}^{\text{opt}} = \arg\max_{\underline{p}} \mathbb{E}\left\{[\underline{p}^t\underline{X}(n)]^2\right\} = \arg\max_{\underline{p}} \underline{p}^t\mathbb{E}\left\{\underline{X}(n)\underline{X}^t(n)\right\}\underline{p} = \arg\max_{\underline{p}} \underline{p}^t\Gamma\underline{p}$$

under the constraint $||\underline{p}|| = 1$. This is a classic problem of principal component analysis (PCA). The derivative of $\underline{p}^t\Gamma\underline{p} - \lambda[\underline{p}^t\underline{p} - 1]$ with respect to $\underline{p}$ is canceled. We find $\Gamma\underline{p} = \lambda\underline{p}$. The parameter λ is therefore an eigenvalue of the matrix Γ. By multiplying through this equation with $\underline{p}^t$, we obtain $\underline{p}^t\Gamma\underline{p} = \lambda\underline{p}^t\underline{p} = \lambda$. The first term is the one that we want to maximize, therefore it is enough to choose the eigenvector associated with the largest eigenvalue of the matrix Γ.

The eigenvalues of the matrix Γ' given by [9.1] are written:

$$\lambda_{1,2} = \frac{1}{2}\left(c + 1/c \pm \sqrt{\Delta}\right) \quad \text{with} \quad \Delta = (c - 1/c)^2 + 4\rho^2(0)$$

The two eigenvectors are orthogonal, therefore if the first (normalized) eigenvector is written $[\cos(\theta)\sin(\theta)]^t$, the second eigenvector is necessarily $[-\sin(\theta)\cos(\theta)]^t$. We therefore have:

$$\begin{bmatrix} c & \rho(0) \\ \rho(0) & 1/c \end{bmatrix} \begin{bmatrix} \cos(\theta) & -\sin(\theta) \\ \sin(\theta) & \cos(\theta) \end{bmatrix} = \begin{bmatrix} \cos(\theta) & -\sin(\theta) \\ \sin(\theta) & \cos(\theta) \end{bmatrix} \begin{bmatrix} \lambda_1 & 0 \\ 0 & \lambda_2 \end{bmatrix}$$

In the following, V denotes the matrix of eigenvectors and Λ is the matrix of eigenvalues.

The largest eigenvalue is:

$$\lambda = \frac{1}{2}\left(c + 1/c + \sqrt{\Delta}\right).$$

If we associate it with the first eigenvector, we obtain:

$$\tan(\theta) = \frac{\rho(0)}{\lambda - 1/c} = \frac{2\rho(0)}{c - 1/c + \sqrt{\Delta}}$$

If we associate it with the second eigenvector, we obtain:

$$\tan(\theta) = \frac{\rho(0)}{c - \lambda} = \frac{2\rho(0)}{c - 1/c - \sqrt{\Delta}}$$

In the two cases, if we make use of the relation:

$$\tan(2\theta) = \frac{2\tan(\theta)}{1 - \tan^2(\theta)}$$

we obtain:

$$\theta = \frac{1}{2}\arctan\left(\frac{2\rho(0)}{c - 1/c}\right)$$

the angle between $-\pi/4$ and $+\pi/4$. When θ is positive, that is, when $\rho(0)$ and $c - 1/c$ have the same sign:

$$s(n) = \cos(\theta)x_1(n) + \sin(\theta)x_2(n)$$

When θ is negative, we must choose:

$$s(n) = -\sin(\theta)x_1(n) + \cos(\theta)x_2(n).$$

It is always more straightforward to calculate the dominant signal using the first relation by choosing:

$$\theta^{\text{opt}} = \mod\left[\frac{1}{2}\arctan\left(\frac{2\rho(0)}{c - 1/c}\right), \frac{\pi}{2}\right] \qquad [9.6]$$

When θ is chosen as $\theta = \theta^{\text{opt}}$, relation [9.5] is simply the Karhunen Loeve transform applied to the vector $\underline{X}(n)$:

$$\begin{bmatrix} S(n) \\ S'(n) \end{bmatrix} = V^t \begin{bmatrix} X_1(n) \\ X_2(n) \end{bmatrix}$$

We have:

$$\mathbb{E}\left\{\begin{bmatrix} S(n) \\ S'(n) \end{bmatrix} [S(n)S'(n)]\right\} = V^t \Gamma V = \sigma_{X_1}\sigma_{X_2}\Lambda$$

The two signals $S(n)$ and $S'(n)$, therefore, have powers $\sigma_S^2 = \sigma_{X_1}\sigma_{X_2}\lambda_1$ and $\sigma_{S'}^2 = \sigma_{X_1}\sigma_{X_2}\lambda_2$ respectively and these are not correlated. We can note that the relationship between the two eigenvalues, that is between the two powers, is given by:

$$\frac{\lambda_2}{\lambda_1} = \frac{c + 1/c - \sqrt{\Delta}}{c + 1/c + \sqrt{\Delta}} = \frac{1 - \sqrt{\mu}}{1 + \sqrt{\mu}} \quad \text{with} \quad \mu = 1 + \frac{4\rho^2(0) - 4}{(c + 1/c)^2} \qquad [9.7]$$

To determine θ^{opt}, another step is possible. In effect, from [9.5] we immediately obtain:

$$\sigma_S^2 = \cos^2(\theta)\sigma_{X_1}^2 + 2\cos(\theta)\sin(\theta)\rho(0)\sigma_{X_1}\sigma_{X_2} + \sin^2(\theta)\sigma_{X_2}^2$$
$$\sigma_{S'}^2 = \sin^2(\theta)\sigma_{X_1}^2 - 2\cos(\theta)\sin(\theta)\rho(0)\sigma_{X_1}\sigma_{X_2} + \cos^2(\theta)\sigma_{X_2}^2$$

or:

$$\tilde{\sigma}_S^2 = \frac{\sigma_S^2}{\sigma_{X_1}\sigma_{X_2}} = c\cos^2(\theta)+2\rho(0)\cos(\theta)\sin(\theta)+\frac{1}{c}\sin^2(\theta) = \frac{1}{2}\left(c+\frac{1}{c}\right)+\frac{y(\theta)}{2}$$
$$\tilde{\sigma}_{S'}^2 = \frac{\sigma_{S'}^2}{\sigma_{X_1}\sigma_{X_2}} = c\sin^2(\theta)-2\rho(0)\cos(\theta)\sin(\theta)+\frac{1}{c}\cos^2(\theta) = \frac{1}{2}\left(c+\frac{1}{c}\right)-\frac{y(\theta)}{2}$$

with:

$$y(\theta) = (c - 1/c)\cos(2\theta) + 2\rho(0)\sin(2\theta)$$

Since:

$$\frac{\partial y(\theta)}{\partial \theta} = -2(c - 1/c)\sin(2\theta) + 4\rho(0)\cos(2\theta)$$

the function $y(\theta)$ has an extremum when:

$$\theta = \frac{1}{2}\arctan\left(\frac{2\rho(0)}{c - 1/c}\right)$$

The four possible cases follow the sign of $\rho(0)$ and of $c-1/c$, which leads to taking θ modulo $\pi/2$ as in equation [9.6]. Knowing θ^{opt}, we can deduce $s_1(n)$ and $s_2(n)$.

9.3.2. *In the frequency domain*

The procedure shown earlier for the time domain can be realized for each tile in the time/frequency decomposition. The angles $\theta^{\text{opt}}(b)$ for $b = 1\cdots B$ must be determined. These depend only on:

$$c(b) = \frac{||\underline{X}_1(b)||}{||\underline{X}_2(b)||}$$

and:

$$\hat{\rho}(0,b) = \frac{1}{2}\frac{\underline{X}_1^t(b)\underline{X}_2^*(b) + \underline{X}_2^t(b)\underline{X}_1^*(b)}{||\underline{X}_1(b)||\;||\underline{X}_2(b)||} = \frac{\Re\left\{\underline{X}_1^t(b)\underline{X}_2^*(b)\right\}}{||\underline{X}_1(b)||\;||\underline{X}_2(b)||}$$
$$\hat{\rho}(0,b) = \frac{|\underline{X}_1^t(b)\underline{X}_2^*(b)|}{||\underline{X}_1(b)||\;||\underline{X}_2(b)||}\cos[\text{ICPD}(b)]$$
$$\hat{\rho}(0,b) = \text{ICC}(b)\;\cos[\text{ICPD}(b)]$$

The three inter-channel indices given in [9.4] are therefore sufficient to calculate the monophonic signal $s(n)$.

9.4. At the receiver

The transmitted bitstream containing the specific information for the monophonic signal $s(n)$ and the three binaural indices is decoded. The same notation for $s(n)$ and the three indices is used in the following although there is necessarily a distortion due to the quantization stage. This distortion is assumed to be negligible in comparison with the approximations which result from the modeling stage. In particular, quantizing the indices is assumed not to denature them (they keep the same perceptual influence).

We saw that at the transmitter, $s(n)$ is characterized by an intermediary rotation for each tile b of the time/frequency decomposition and for each component $k \in \{k_b \cdots k_{b+1} - 1\}$ of the b^{th} critical band by writing:

$$\begin{bmatrix} S(k) \\ S'(k) \end{bmatrix} = \begin{bmatrix} \cos(\theta^{\text{opt}}(b)) & \sin(\theta^{\text{opt}}(b)) \\ -\sin(\theta^{\text{opt}}(b)) & \cos(\theta^{\text{opt}}(b)) \end{bmatrix} \begin{bmatrix} X_1(k) \\ X_2(k) \end{bmatrix}$$

or, to simplify the notation:

$$\begin{bmatrix} S \\ S' \end{bmatrix} = \begin{bmatrix} \cos(\theta) & \sin(\theta) \\ -\sin(\theta) & \cos(\theta) \end{bmatrix} \begin{bmatrix} X_1 \\ X_2 \end{bmatrix}$$

At the receiver, the inverse operation must be performed:

$$\begin{bmatrix} \hat{X}_1 \\ \hat{X}_2 \end{bmatrix} = \begin{bmatrix} \cos(\theta) & -\sin(\theta) \\ \sin(\theta) & \cos(\theta) \end{bmatrix} \begin{bmatrix} S \\ S' \end{bmatrix}$$

If S' or $s'(n)$ is known, we obtain[5] $\hat{X}_1 = X_1$ and $\hat{X}_2 = X_2$. Since they are not known, several modifications are introduced.

9.4.1. *Stereo signal reconstruction*

This involves obtaining a signal which can be substituted for $s'(n)$. We are first of all interested in the *downmixing* procedure and looking for a physical interpretation in the case of a stereophonic restitution. The two signals $x_1(n)$ and $x_2(n)$ are assumed to be supplied to two loud-speakers placed at the vertices A and B of an isosceles triangle characterized by the angle BOx = xOA = ψ_0 as shown in the diagram in Figure 9.2. Conventionally, the signal $x_1(n)$ corresponds to the left channel and $x_2(n)$ to the right channel. The third vertex O corresponds to the position of the listener (*sweet spot*). The two channels create a virtual source located at M which is characterized by the angle xOM equal to ψ, the azimuth of the source.

5. The conditions for the overlap-add synthesis must be added.

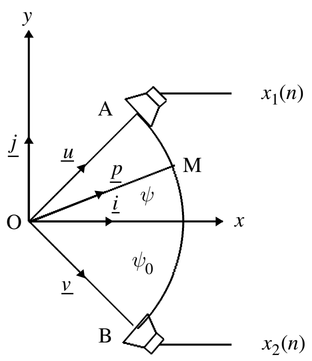

Figure 9.2. *Stereophonic restitution*

Noticing that $\psi = \psi_0$ when c is very high (the virtual source is at A) and $\psi = -\psi_0$ when c is very small (the virtual source is at B), we obtain:

$$\psi = \psi_0 \left(-\frac{4}{\pi}\theta + 1 \right)$$

To a first approximation, in the case of a single sound source, we can see that the signal $s(n)$ specifies this virtual source and that the signal $s'(n)$ represents the "orthogonal" information which conveys the reverberation effect. Since the reverberation is known to be composed of a large number of delayed and attenuated versions of the source, we can see that $s'(n)$ must be a decorrelated version of $s(n)$ but with the same spectral-temporal envelope and that the most basic way is to take a simple, delayed version or to filter it with a comb filter and by a all-pass filter. In [BRE 05], Breebaart *et al.* put forward the use of an filter impulse reponse:

$$h(n) = \frac{2}{N} \sum_{k=0}^{N/2} \cos \left(2\pi \frac{k}{N} n + 2\pi \frac{k}{N}(k-1) \right)$$

with $N = 640$. We can show that the amplitude spectrum is flat and that the phase spectrum is very jerky. We can also verify that it has a power gain of 1.

9.4.2. *Power adjustment*

We have seen that S and S' at the transmitter do not have the same power at all. If at the receiver S' is deduced from S, we can assume that we can create a component with the same power. It is therefore necessary to adjust the powers to an acceptable level. If we write:

$$\begin{bmatrix} \hat{S} \\ \hat{S}' \end{bmatrix} = \begin{bmatrix} \cos(\varphi) & 0 \\ 0 & \sin(\varphi) \end{bmatrix} \begin{bmatrix} S \\ S' \end{bmatrix}$$

we can see that this parameterization leads to the preservation of the powers $\sigma^2_{\hat{S}} + \sigma^2_{\hat{S}'} = \sigma^2_S$. Following this, it is enough to ensure that the ratio of the powers is equal to the ratio of the eigenvalues given by [9.7]:

$$\tan(\varphi) = \frac{\lambda_2}{\lambda_1} = \frac{1-\sqrt{\mu}}{1+\sqrt{\mu}} \quad \text{with} \quad \mu = 1 + \frac{4\rho^2(0) - 4}{(c+1/c)^2}$$

9.4.3. *Phase alignment*

A *phase alignment* is introduced as proposed by Breebaart *et al.* [BRE 05] in adding a supplementary matrix in the form:

$$\begin{bmatrix} \hat{X}_1 \\ \hat{X}_2 \end{bmatrix} = \begin{bmatrix} \exp(j\Phi_1) & 0 \\ 0 & \exp(j\Phi_2) \end{bmatrix} \begin{bmatrix} \cos(\theta) & -\sin(\theta) \\ \sin(\theta) & \cos(\theta) \end{bmatrix} \begin{bmatrix} \hat{S} \\ \hat{S}' \end{bmatrix}$$

Let us give an interpretation of Φ_1 and Φ_2 in the complex plane. We can note first of all that interpretation in the complex plane can only be done with (complex) scalars and not vectors when we want to interpret the behavior of a certain number of components in a sub-band. In reality, a specific average behavior is interpreted in each sub-band.

Let us now go to the transmitter and assume that the vectors $\underline{X}_1(b)$ and $\underline{X}_2(b)$ can be represented by two complex numbers X_1 and X_2. Consider the representation given in Figure 9.3.

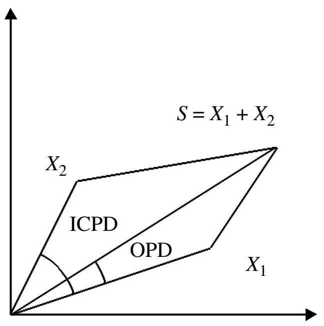

Figure 9.3. *In the complex plane: the case where* $S = X_1 + X_2$

Referring to the images of the two complex numbers X_1 and X_2 as M_1 and M_2, the inter-channel phase difference:

$$\text{ICPD}(b) = \arg\{\underline{X}_1^t(b)\, \underline{X}_2^*(b)\}$$

can be interpreted as the angle $(\mathrm{Ox}, \mathrm{OM}_1) - (\mathrm{Ox}, \mathrm{OM}_2) = -(\mathrm{OM}_1, \mathrm{OM}_2)$. We introduce the *Overall Phase Difference*:

$$\mathrm{OPD}(b) = \arg\{\underline{X}_1^t(b)\, \underline{S}^*(b)\}$$

which has an interpretation in the complex plane given in Figure 9.3. We can see that if we set $\Phi_1 = \mathrm{OPD}$ and $\Phi_2 = \mathrm{OPD} - \mathrm{ICPD}$, we obtain $\hat{X}_1$ and $\hat{X}_2$ at the receiver aligned "on average" with X_1 and X_2 at the transmitter.

Knowing $\hat{X}_1$ and $\hat{X}_2$, two overlap-add syntheses, one for each channel, enables the calculation of $\hat{x}_1(n)$ and $\hat{x}_2(n)$.

9.4.4. *Information transmitted via the channel*

For each tile of the time/frequency decomposition, the ICLD, ICPD, ICC, and *a priori* the OPD must be transmitted. In reality, transmitting the OPD is not necessary.

In effect, if we realize a passive *downmixing* $S = X_1 + X_2$, Lapierre and Lefebvre [LAP 06] have shown that:

$$\mathrm{OPD} = \arg\left\{\underline{X}_1^t\,(\underline{X}_1^* + \underline{X}_2^*)\right\} = \arg\left\{||\underline{X}_1||^2 + |\underline{X}_1^t\,\underline{X}_2^*|\exp(j\ \mathrm{ICPD})\right\}$$

$$\mathrm{OPD} = \arg\left\{\frac{||\underline{X}_1||}{||\underline{X}_2||} + \frac{|\underline{X}_1^t\,\underline{X}_2^*|}{||\underline{X}_1||\,||\underline{X}_2||}\exp(j\ \mathrm{ICPD})\right\}$$

$$\mathrm{OPD} = \arg\{c + \mathrm{ICC}\exp(j\ \mathrm{ICPD})\}$$

This result can be immediately generalized to the case of an active *downmix*. In effect:

$$\mathrm{OPD} = \arg\left\{\underline{X}_1^t\,[\cos(\theta)\underline{X}_1^* + \sin(\theta)\underline{X}_2^*]\right\}$$

$$\mathrm{OPD} = \arg\left\{\cos(\theta)\frac{||\underline{X}_1||}{||\underline{X}_2||} + \sin(\theta)\frac{|\underline{X}_1^t\,\underline{X}_2^*|}{||\underline{X}_1||\,||\underline{X}_2||}\exp(j\mathrm{ICPD})\right\}$$

$$\mathrm{OPD} = \arg\{\cos(\theta)\ c + \sin(\theta)\ \mathrm{ICC}\exp(j\ \mathrm{ICPD})\}$$

9.5. Draft International Standard

The document [INT 05] shows that in reality there is no active *downmix*. The monophonic signal is directly given by $s(n) = [x_1(n) + x_2(n)]/2$.

In Table 9.1, the quantization code books for the ICLD, ICPD, and ICC parameters are presented. Apparently, the OPD is also quantized with the ICPD code book and transmitted. Under the hypothesis in which only the first three parameters are

ICLD	-25	-18	-14	-10	-7	-4	-2	0	2
	4	7	10	14	18	25			
ICPD	0	$\pi/4$	$\pi/2$	$3\pi/4$	π	$5\pi/4$	$3\pi/2$	$7\pi/4$	2π
ICC	1	0.937	0.841	0.6	0.367	0	-0.589	-1	

Table 9.1. *Quantization codebook for ICLD, ICPD, and OPD*

transmitted, this has a cost of 10 bits per analysis frame. If we transfer these parameters every 20 ms, the corresponding bit rate is 2 kbit/s.

In Figure 9.4, the angles θ, calculated at the receiver as a function of time, are shown for the critical bands when the two channels $x_1(n)$ and $x_2(n)$ of the stereophonic signal processed are created from a unique source $s(n)$ as follows: $x_1(n) = g_1 \times s(n)$ and $x_2(n) = g_2 \times s(n)$. Here, we have simultaneously a violin appearing to move from the left-hand loudspeaker toward the right-hand loudspeaker and a piano which appears to do the opposite.

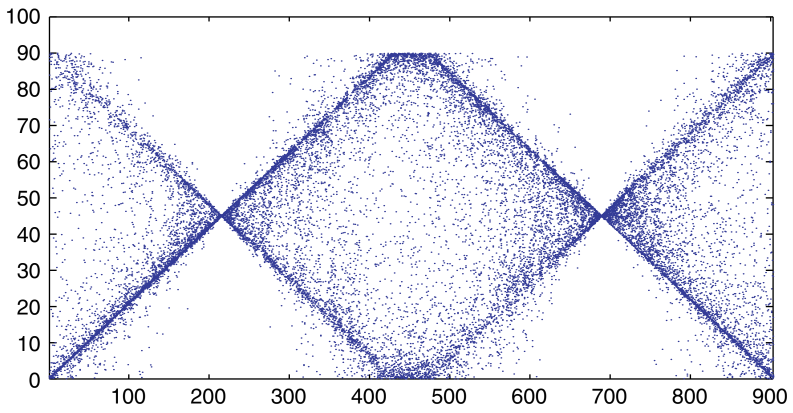

Figure 9.4. *θ angles as a function of time calculated for the set of critical bands*

PART 3

MATLAB® Programs

Chapter 10

A Speech Coder

10.1. Introduction

The program presented in this chapter does not, strictly speaking, simulate a speech coder. From the file of a telephone-band speech signal (F_e = 8 kHz) in wave format (quantized on 16 bits), it does not construct a file containing code words to transmit (the *bitstream*). It does the modeling stage which consists of determining the synthesis filter parameters and the characterizing parameters for the input signal to this filter for each analysis frame. To simulate a real coder, the parameter quantization stage must be realized and the parts realized by the encoder and decoder must be realized separately. Since a speech coder obeys a synthesis analysis scheme, the program simulating the decoder is reduced to a small sub-set of the whole program (mostly decoding the code words).

With the values suggested for the analysis frame length, the synthesis filter order, and the code book size, this program simulates a coder with a bit rate of the order of 10 kb/s.

10.2. Script for the calling function

```
clear

% Parameters
Fe = 8000;
N = 320;
M = 160;
P = 12;
```

```
Num_sfrm = 4;
M_prime = M/Num_sfrm;
L_plt = 256;
L = 512;
Dico_C = randn(M/Num_sfrm,L);
Num_vectors = 2;
Gamma = 0.8;

% Bit rate estimation
Num_bits_a = 30;
Num_bits_gain = 5;
D = Num_bits_a*Fe/M;
D = D + Num_sfrm*Num_vectors*(Num_bits_gain+log2(L))*Fe/M;
D = D + Num_sfrm*(Num_bits_gain+log2(L_plt))*Fe/M;
fprintf('Bit rate = %d kbits/s\n', round(D/1000))
fprintf('Gamma = %.2f\n', Gamma)

% Reading the signal
prefix = 'C:\Documents and Settings\Nicolas\My documents';
prefix_in  = [prefix '\my_signals\signals_ref\fe_8\'];
prefix_out = [prefix '\my_signals\signals_temp\'];
signal_xn = wavread([prefix_in 'voice_female_1.wav'], [0.15 3]*Fe);
Num_sam = length(signal_xn);

Num_sam_visu = Fe/4;
axis_time = (0:Num_sam_visu-1)'/Fe;
vmax_xn = max(abs(signal_xn));
figure(1); hold off
plot(axis_time, signal_xn(1:Num_sam_visu)); hold on
axis([0 Num_sam_visu/Fe 1.1*max(abs(signal_xn))*[-1 1]])

% Some reservations
state_yn = zeros(P,1);
state_pn = zeros(P,1);
state_ringing = zeros(P,1);
state_xn_hat = zeros(P,1);
signal_yn = zeros(Num_sam, 1);
signal_yn_hat = zeros(Num_sam, 1);
signal_xn_hat = zeros(Num_sam, 1);
buffer_yn_hat = zeros(L_plt+M_prime,1);
vn = 0.56 - 0.44*cos(2*pi*(0:N-1)'/(N-1));

% Processing by frame
Num_frm = fix((Num_sam-N)/M) + 1;
Num_frm_visu = fix((Num_sam_visu-N)/M) + 1;
for no_frm = 1:Num_frm
   n1 = (no_frm-1)*M + 1;
```

```
n2 = n1 + N - 1;
n3 = n1 + (N-M)/2;
n4 = n3 + M - 1;

% LPC analysis and calculating the residual signal
xn = vn.*signal_xn(n1:n2);
[a sigma2] = analyse_lpc(xn, P);
xn = signal_xn(n3:n4);
[yn, state_yn] = filtre_ma(a, xn, state_yn);
signal_yn(n3:n4) = yn;

% Calculating the perceptual signal
a_percep = a.*(Gamma.^(0:P)');
[pn, state_pn] = filtre_ar(1, a_percep, yn, state_pn);

% Filtering the "stochastic" code book
Dico_F = zeros(size(Dico_C));
alpha = zeros(L,1);
for i = 1:L
    Dico_F(:,i) = filter(1, a_percep, Dico_C(:,i));
    alpha(i) = norm(Dico_F(:,i));
end

% Determining the synthesis filter input
yn_hat = zeros(M,1);
for no_sfrm = 1:Num_sfrm
    n5 = (no_sfrm-1)*M_prime + 1;
    n6 = no_sfrm*M_prime;

    % Constructing Dico_C_plt from buffer_yn_hat
    Dico_C_plt = zeros(M_prime, L_plt);
    for i = 1:L_plt
        Dico_C_plt(:,i) = buffer_yn_hat(i:i+M_prime-1);
    end
    % Filtering the code book
    Dico_F_plt = zeros(M_prime,L_plt);
    alpha_plt = zeros(L_plt,1);
    for i = 1:L_plt
        Dico_F_plt(:,i) = filter(1, a_percep, Dico_C_plt(:,i));
        alpha_plt(i) = norm(Dico_F_plt(:,i));
    end

    % Finding the 1st vector in the adaptive code book
    ringing = filtre_ar(1, a_percep, zeros(M_prime,1), state_ringing);
    en = pn(n5:n6) - ringing;
    [gain_plt, indice_plt] = determination_gains_shapes( ...
        en, Dico_F_plt, alpha_plt, 1);
        yn_hat(n5:n6) = yn_hat(n5:n6) + ...
            gain_plt*Dico_C_plt(:,indice_plt);
        en = en - gain_plt*Dico_F_plt(:,indice_plt);

        % Looking again in the stochastic code book
        [gains, indices] = determination_gains_shapes( ...
            en, Dico_F, alpha, Num_vectors);
        for k = 1:Num_vectors
            yn_hat(n5:n6) = yn_hat(n5:n6) + ...
                gains(k)*Dico_C(:,indices(k));
        end

        % Updating the buffer while safeguarding yn_hat
        for l = 1:L_plt
            buffer_yn_hat(l) = buffer_yn_hat(l+M_prime);
        end
        buffer_yn_hat(L_plt+1:L_plt+M_prime) = yn_hat(n5:n6);

        % Updating the ringing
        [ans, state_ringing] = filtre_ar(1, a_percep, yn_hat(n5:n6), ...
            state_ringing);
    end
    signal_yn_hat(n3:n4) = yn_hat;

    % Calculating xn_hat
    [xn_hat, state_xn_hat] = filtre_ar(1, a, yn_hat, state_xn_hat);
    signal_xn_hat(n3:n4) = xn_hat;

    % Visualizations
    if no_frm <= Num_frm_visu
        figure(1);
        plot(axis_time(n1:n2), vmax_xn*vn, 'r');
        plot(axis_time(1:n4), signal_xn_hat(1:n4), 'r');
        plot(axis_time(n3)*[1 1], 1.1*vmax_xn*[-1 1], 'g')
        plot(axis_time(n4)*[1 1], 1.1*vmax_xn*[-1 1], 'g')

        figure(3); hold off
        plot(xn); hold on
        plot(xn_hat, 'r')
        plot(xn-xn_hat, 'k')
        plot([1 M], [0 0])
        for no_sfrm = 1:Num_sfrm
            n5 = (no_sfrm-1)*M_prime + 1;
            plot(n5*[1 1], 1.1*max(abs(xn))*[-1 1], 'g')
        end
        axis([1 M 1.1*max(abs(xn))*[-1 1]])
        % Visualizing the spectra for the preceding analysis frame
        off_set = 90;
        axis_freq = (0:N/2-1)*Fe/N/1000;
        if no_frm > 1
            n1 = (no_frm-2)*M + 1;
            n2 = n1 + N - 1;

            signal_weight = vn.*signal_xn(n1:n2);
            time = (abs(fft(signal_weight)).^2)/N;
            perio_xn_db = 10*log10(time(1:N/2)) + off_set;
            [a_bis sigma2_bis] = analyse_lpc(signal_weight, P);
            time = abs(fft(a_bis,N)).^2;
            spectre_lpc_xn_db = 10*log10(sigma2_bis./time(1:N/2)) ...
                + off_set;

            signal_weight = vn.*signal_xn_hat(n1:n2);
            time = (abs(fft(signal_weight)).^2)/N;
            perio_xn_hat_db = 10*log10(time(1:N/2)) + off_set;
            [a_bis sigma2_bis] = analyse_lpc(signal_weight, P);
            time = abs(fft(a_bis,N)).^2;
            spectre_lpc_xn_hat_db = 10*log10(sigma2_bis./time(1:N/2)) ...
                + off_set;

            signal_weight = vn.*(signal_xn(n1:n2)-signal_xn_hat(n1:n2));
            time = (abs(fft(signal_weight)).^2)/N;
            perio_in_db = 10*log10(time(1:N/2)) + off_set;

            figure(2); hold off
            plot(axis_freq, perio_xn_db, 'linewidth', 2); hold on
            plot(axis_freq, perio_xn_hat_db, 'r');
            plot(axis_freq, perio_in_db, 'k');
            axis([0 Fe/2000 0 100])
        end
        drawnow
        fprintf('pause \n'); pause
    else
        if rem(no_frm,20) ==0
            fprintf('no_frm = %3d/%3d\n', no_frm, Num_frm)
        end
    end
end

wavwrite(signal_xn, Fe, [prefix_out 'voice_original']);
wavwrite(signal_xn_hat, Fe, ...
   [prefix_out 'coder_celp_with_plt_gamma']);
```

10.3. Script for called functions

```
function [a, sigma2] = analyse_lpc(xn, P)
N = length(xn);
rk = zeros(P+1,1);
for k = 1:P+1
   rk(k) = xn(1:N-k+1)'*xn(k:N)/N;
end
if rk(1) > 0
   R = toeplitz(rk(1:P));
   a = [1; -inv(R)*rk(2:P+1)];
   sigma2 = a'*rk;
else
   a = [1; zeros(P,1)];
   sigma2 = 0;
end

function [yn, state] = filter_ma(bi, xn, state)
P = length(state);
yn = zeros(size(xn));
for n = 1:length(xn)
   yn(n) = bi(1)*xn(n) + bi(2:P+1)'*state;
   state(2:P) = state(1:P-1);
   state(1) = xn(n);
end

function [yn, state] = filter_ar(b0, ai, xn, state)
P = length(state);
yn = zeros(size(xn));
for n = 1:length(xn)
   yn(n) = b0*xn(n) - ai(2:P+1)'*state;
   state(2:P) = state(1:P-1);
   state(1) = yn(n);
end

function [gains, indices] = determination_gains_shapes( ...
   en, Dico, alpha, Num_vectors)
[M, L] = size(Dico);
gains = zeros(Num_vectors,1);
indices = ones(Num_vectors,1);
for k = 1:Num_vectors
   criterion_max = 0;
   for j = 1:L
       beta = Dico(:,j)'*en;
       if alpha(j) ~= 0
           criterion = abs(beta)/alpha(j);
       else
```

```
            criterion = 0;
        end
        if criterion > criterion_max
            gains(k) = beta/(alpha(j)^2);
            indices(k) = j;
            criterion_max = criterion;
        end
    end
    en = en - gains(k)*Dico(:,indices(k));
end
```

Chapter 11

A Music Coder

11.1. Introduction

The program presented in this chapter simulates the MPEG-1 Layer-1 coder with a music signal sampled at 44.1 kHz at a bit rate between 64 and 96 kb/s. The quantization tables provided are only for this specific sampling frequency and bit rate range. In contrast with the program for the speech coder simulation, the program in this chapter does indeed construct an intermediate file containing the code words. However, the exact syntax given in the standardization document is not observed in this program.

11.2. Script for the calling function

```
clear all
global Fe N_mpeg M_mpeg D_mpeg

Fe = 44100;
N_mpeg = 512;
M_mpeg = 32;
D_mpeg = 12;
Bitrate_kbs = 64;

% Reading the signal
prefix = 'C:\Documents and Settings\Nicolas\My documents';
prefix_in  = [prefix '\my_signals\signals_ref\fe_44.1\'];
prefix_out = [prefix '\my_signals\signals_temp\'];
name = 'partita_no3';
signal_xn = wavread([prefix_in name], [1.7 2.7]*Fe);
```

```
Num_sam = length(signal_xn);
signal_xn = (signal_xn(:,1)+signal_xn(:,2))/2;

figure(1); hold off
plot((1:Num_sam)/Fe, signal_xn); hold on
xlabel('Time [s]')
ylabel ('Amplitude')
axis([0 Num_sam/Fe 1.1*max(abs(signal_xn))*[-1 1]]);
drawnow

% Some initialisations
[hn, Mat_H] = init_bank_pqmf;
tables_quantiz;
init_mpa1;

% Encoding
fprintf('Encoding at %d kbits/s from file %s \n', Bitrate_kbs, name)
fid_bit_stream = fopen('bit_stream', 'w');
xn  = zeros(N_mpeg, 1);
xn_mpa = zeros(N_mpeg, 1);
Mat_sb = zeros(D_mpeg, M_mpeg);
hann = sqrt(8/3)/2*(ones(N_mpeg, 1) - cos(2*pi*(0:N_mpeg-1)'/N_mpeg));
Delay_yn  = N_mpeg + (D_mpeg-1)*M_mpeg;
Delay_mpa = N_mpeg + (D_mpeg-1)*M_mpeg/2;
no_sam_sb = 1;
Num_blocs = fix(Num_sam/M_mpeg);
for no_bloc = 1:Num_blocs
   % Updating xn and xn_mpa
   n1 = (no_bloc-1)*M_mpeg + 1;
   n2 = n1 + M_mpeg - 1;
   xn(1:N_mpeg-M_mpeg) = xn(M_mpeg+1:N_mpeg);
   xn(N_mpeg-M_mpeg+1:N_mpeg) = signal_xn(n1:n2);
   xn_mpa(1:N_mpeg-M_mpeg) = xn_mpa(M_mpeg+1:N_mpeg);
   ind = 2*N_mpeg-Delay_mpa-M_mpeg+1:2*N_mpeg-Delay_mpa;
   xn_mpa(N_mpeg-M_mpeg+1:N_mpeg) = xn(ind);

   % Bank of analysis filters
   Mat_sb(no_sam_sb,:) = (Mat_H*xn)';
   no_sam_sb = rem(no_bloc,D_mpeg) + 1;

   % Every 12 samples in the sub-band
   if rem(no_bloc, D_mpeg) == 0
       % Calculating the masking threshold
       [thresh1_db, thresh2_db, SMR_db_sb] = mpeg_mpa1(xn_mpa);
       % Bit allocation
       NOQ = mpeg_alloc(SMR_db_sb, Bitrate_kbs);
```

```
        % Quantizing and constructing the bitstream
        write_bit_stream(fid_bit_stream, NOQ, Mat_sb)

        % Some graphics
        off_set = 90;
        figure(2); hold off
        time = fft(xn_mpa.*hann);
        X1 = 10*log10((abs(time(1:N_mpeg/2)).^2)/N_mpeg);
        axe_freq = (0:N_mpeg/2-1)*Fe/1000/N_mpeg;
        plot(axis_freq, X1+off_set); hold on
        plot(axis_freq, thresh1_db+off_set, 'r');
        plot(axis_freq, thresh2_db+off_set, 'g');
        val_old = SMR_db_sb(1);
        for k = 1:M_mpeg
            ind = (k-1)*8+1:k*8+1;
            plot(ind(1)*Fe/1000/N_mpeg*[1 1], [val_old SMR_db_sb(k)])
            plot(ind*Fe/1000/N_mpeg, SMR_db_sb(k)*ones(9,1))
            val_old = SMR_db_sb(k);
        end
        plot([0 Fe/2], [0 0], ':');
        xlabel('Frequency [kHz]');
        ylabel('Power [dB]');
        axis([0 Fe/1000/2 -20 100]);
        drawnow
    end
    if rem(no_bloc,100) == 0
        fprintf(' %4.2f', no_bloc*M_mpeg/Fe)
        if rem(no_bloc,1000) == 0
            fprintf('\n')
        end
    end
end
fprintf('\n')
fclose(fid_bit_stream);

% Decoding
fprintf('Decoding\n')
fid_bit_stream = fopen('bit_stream');
signal_xn_hat = zeros(Num_sam,1);
xn_hat = zeros(N_mpeg,1);
Mat_sb_q = zeros(D_mpeg,M_mpeg);
Num_blocs_deb_transf = N_mpeg/M_mpeg + D_mpeg;
no_bloc = 0;
no_sam_sb = 1;
while 1
    no_bloc = no_bloc + 1;
```

```
   % Synthesis filter bank
   xn_hat = xn_hat + Mat_H'*Mat_sb_q(no_sam_sb,:)';
   no_sam_sb = rem(no_bloc,D_mpeg) + 1;

   if no_bloc >= Num_blocs_deb_transf
       n1 = (no_bloc-Num_blocs_deb_transf)*M_mpeg + 1;
       n2 = n1 + M_mpeg - 1;
       signal_xn_hat(n1:n2) = xn_hat(1:M_mpeg);
   end
   xn_hat(1:N_mpeg-M_mpeg) = xn_hat(M_mpeg+1:N_mpeg);
   xn_hat(N_mpeg-M_mpeg+1:N_mpeg) = zeros(M_mpeg,1);

   if rem(no_bloc,D_mpeg) == 0
       [Mat_sb_q, stop] = lit_bit_stream(fid_bit_stream);
       if stop
           fprintf('\n')
           break
       end
   end
   if rem(no_bloc,100) == 0
       fprintf(' %4.2f', no_bloc*M_mpeg/Fe)
       if rem(no_bloc,1000) == 0
           fprintf('\n')
       end
   end
end
fclose(fid_bit_stream);

wavwrite(signal_xn, Fe, [prefix_out 'signal_original']);
wavwrite(signal_xn_hat, Fe, [prefix_out 'signal_reconstructed']);
wavwrite((signal_xn-signal_xn_hat(1:Num_sam)), ...
   Fe, [prefix_out 'diff']);

figure(1);
plot((1:Num_sam)/Fe, signal_xn-signal_xn_hat(1:Num_sam), 'r');
```

11.3. Script for called functions

```
function [hn, Mat_H] = init_bank_pqmf

% Defining the impulse response of the prototype filter
% and the modulation matrix for the PQMF filter bank

global N_mpeg M_mpeg
```

```
time = [ 0.202407  0.202283  0.201916  0.201307 ...
         0.200452  0.199359  0.198029  0.196465 ...
         0.194668  0.192648  0.190409  0.187952 ...
         0.185290  0.182422  0.179361  0.176113 ...
         0.172685  0.169084  0.165321  0.161407 ...
         0.157347  0.153153  0.148837  0.144402 ...
         0.139868  0.135239  0.130527  0.125745 ...
         0.120900  0.116004  0.111068  0.106105 ...
         0.101123  0.096135  0.091148  0.086174 ...
         0.081224  0.076307  0.071432  0.066610 ...
         0.061849  0.057155  0.052540  0.048011 ...
         0.043576  0.039242  0.035012  0.030899 ...
         0.026907  0.023036  0.019297  0.015693 ...
         0.012227  0.008901  0.005724  0.002692 ...
        -0.000189 -0.002919 -0.005495 -0.007917 ...
        -0.010185 -0.012303 -0.014264 -0.016074 ...
        -0.017733 -0.019243 -0.020608 -0.021827 ...
        -0.022906 -0.023845 -0.024652 -0.025326 ...
        -0.025873 -0.026300 -0.026604 -0.026799 ...
        -0.026882 -0.026863 -0.026747 -0.026537 ...
        -0.026238 -0.025855 -0.025399 -0.024867 ...
        -0.024271 -0.023616 -0.022904 -0.022143 ...
        -0.021336 -0.020492 -0.019613 -0.018706 ...
        -0.017773 -0.016824 -0.015858 -0.014882 ...
        -0.013900 -0.012915 -0.011936 -0.010960 ...
        -0.009994 -0.009039 -0.008103 -0.007183 ...
        -0.006285 -0.005411 -0.004564 -0.003744 ...
        -0.002954 -0.002196 -0.001470 -0.000777 ...
        -0.000121  0.000499  0.001084  0.001632 ...
         0.002142  0.002616  0.003051  0.003453 ...
         0.003814  0.004141  0.004435  0.004691 ...
         0.004915  0.005106  0.005265  0.005395 ...
         0.005495  0.005565  0.005611  0.005629 ...
         0.005624  0.005597  0.005549  0.005481 ...
         0.005397  0.005292  0.005176  0.005044 ...
         0.004901  0.004745  0.004580  0.004408 ...
         0.004227  0.004041  0.003852  0.003658 ...
         0.003461  0.003264  0.003067  0.002870 ...
         0.002673  0.002479  0.002287  0.002101 ...
         0.001918  0.001740  0.001567  0.001400 ...
         0.001238  0.001082  0.000936  0.000793 ...
         0.000658  0.000531  0.000413  0.000299 ...
         0.000194  0.000097  0.000005 -0.000078 ...
        -0.000154 -0.000224 -0.000286 -0.000343 ...
        -0.000394 -0.000440 -0.000477 -0.000510 ...
        -0.000539 -0.000561 -0.000580 -0.000596 ...
        -0.000604 -0.000612 -0.000615 -0.000615 ...
        -0.000612 -0.000607 -0.000599 -0.000588 ...
        -0.000575 -0.000561 -0.000545 -0.000529 ...
        -0.000513 -0.000494 -0.000475 -0.000456 ...
        -0.000434 -0.000415 -0.000397 -0.000375 ...
        -0.000356 -0.000337 -0.000316 -0.000299 ...
        -0.000281 -0.000262 -0.000245 -0.000229 ...
        -0.000213 -0.000197 -0.000183 -0.000170 ...
        -0.000156 -0.000143 -0.000132 -0.000121 ...
        -0.000111 -0.000103 -0.000094 -0.000084 ...
        -0.000078 -0.000070 -0.000065 -0.000057 ...
        -0.000051 -0.000046 -0.000043 -0.000038 ...
        -0.000035 -0.000030 -0.000027 -0.000024 ...
        -0.000022 -0.000019 -0.000019 -0.000016 ...
        -0.000013 -0.000013 -0.000011 -0.000011 ...
        -0.000008 -0.000008 -0.000005 -0.000005 ...
        -0.000005 -0.000005 -0.000003 -0.000003 ...
        -0.000003 -0.000003 -0.000003 -0.000003];
hn = [0 fliplr(time(2:256)) time]';

Mat_H = zeros(M_mpeg, N_mpeg);
for k = 0:M_mpeg-1
    t2 = ((2*k+1)*pi/(2*M_mpeg))*((0:N_mpeg-1)+M_mpeg/2);
    Mat_H(k+1,:) = hn'.*cos(t2);
end

function init_tables_quantiz

% Initializing the quantization tables

global Scl_fact Rsb_iso_1_3 Rsb_iso_4_11 Rsb_iso_12_23 Rsb_iso_24_27

% Scale factors
Scl_fact = [2.00000000000000, 1.58740105196820, 1.25992104989487, ...
            1.00000000000000, 0.79370052598410, 0.62996052494744, ...
            0.50000000000000, 0.39685026299205, 0.31498026247372, ...
            0.25000000000000, 0.19842513149602, 0.15749013123686, ...
            0.12500000000000, 0.09921256574801, 0.07874506561843, ...
            0.06250000000000, 0.04960628287401, 0.03937253280921, ...
            0.03125000000000, 0.02480314143700, 0.01968626640461, ...
            0.01562500000000, 0.01240157071850, 0.00984313320230, ...
            0.00781250000000, 0.00620078535925, 0.00492156660115, ...
            0.00390625000000, 0.00310039267963, 0.00246078330058, ...
            0.00195312500000, 0.00155019633981, 0.00123039165029, ...
            0.00097656250000, 0.00077509816991, 0.00061519582514, ...
            0.00048828125000, 0.00038754908495, 0.00030759791257, ...
            0.00024414062500, 0.00019377454248, 0.00015379895629, ...
            0.00012207031250, 0.00009688727124, 0.00007689947814, ...
            0.00006103515625, 0.00004844363562, 0.00003844973907, ...
            0.00003051757813, 0.00002422181781, 0.00001922486954, ...
            0.00001525878906, 0.00001211090890, 0.00000961243477, ...
            0.00000762939453, 0.00000605545445, 0.00000480621738, ...
            0.00000381469727, 0.00000302772723, 0.00000240310869, ...
            0.00000190734863, 0.00000151386361, 0.00000120155435];

% Signal-to-noise ratio as a function of the number of quantization steps
% The third column contains the number of bits necessary
% For n=3,5,7,9 data are grouped in threes: 3 => ceil(log2(n^3))/3

% for sub-bands 0,1 et 2
Rsb_iso_1_3 = [0  0.00 0
               3  7.00 1.67
               7 16.00 3
              15 25.28 4
              31 31.59 5
              63 37.75 6
             127 43.84 7
             255 49.89 8
             511 55.93 9
            1023 61.96 10
            2047 67.98 11
            4095 74.01 12
            8191 80.03 13
           16383 86.05 14
           32767 92.01 15
           65535 98.01 16];

% for sub-bands 3 to 10
Rsb_iso_4_11 = [0  0.00 0
                3  7.00 1.67
                5 11.00 2.33
                7 16.00 3
                9 20.84 3.33
               15 25.28 4
               31 31.59 5
               63 37.75 6
              127 43.84 7
              255 49.89 8
              511 55.93 9
             1023 61.96 10
             2047 67.98 11
             4095 74.01 12
             8191 80.03 13
            65535 98.01 16];
```

```
% for sub-bands 11 to 22
Rsb_iso_12_23 = [0   0.00 0
                 3   7.00 1.67
                 5  11.00 2.33
                 7  16.00 3
                 9  20.84 3.33
                15  25.28 4
                31  31.59 5
             65535  98.01 16];

% for sub-bands 23 à 26
Rsb_iso_24_27 = [0   0.00 0
                 3   7.00 1.67
                 5  11.00 2.33
             65535  98.01 16];

A_B = [3 0.750000000 -0.250000000
       7 0.875000000 -0.125000000
      15 0.937500000 -0.062500000
      31 0.968750000 -0.031250000
      63 0.984375000 -0.015625000
     127 0.992187500 -0.007812500
     255 0.996093750 -0.003906250
     511 0.998046875 -0.001953125
    1023 0.999023438 -0.000976563
    2047 0.999511719 -0.000488281
    4095 0.999755859 -0.000244141
    8191 0.999877930 -0.000122070
   16383 0.999938965 -0.000061035
   32767 0.999969482 -0.000030518];

function init_mpa1

% Defining constants used in mpeg psychoacoustic model 1

global Fe LTq_i LTq_k Table_z Frontieres_i Frontieres_k Larg_f

% Determining the "k" indices as a function of the "i" indices
i1 = 1:48;
i2 = (49:72)-48;
i3 = (73:108)-72;
i_to_k = [i1 (2*i2+48) (4*i3+96)];

% Absolute hearing threshold as a function of the "i" indices
LTq_i = [25.87 14.85 10.72  8.50  7.10  6.11  5.37  4.79 ...
          4.32  3.92  3.57  3.25  2.95  2.67  2.39  2.11 ...
          1.83  1.53  1.23  0.90  0.56  0.21 -0.17 -0.56 ...
```

```
         -0.96 -1.38 -1.79 -2.21 -2.63 -3.03 -3.41 -3.77 ...
         -4.09 -4.37 -4.60 -4.78 -4.91 -4.97 -4.98 -4.92 ...
         -4.81 -4.65 -4.43 -4.17 -3.87 -3.54 -3.19 -2.82 ...
         -2.06 -1.32 -0.64 -0.04  0.47  0.89  1.23  1.51 ...
          1.74  1.93  2.11  2.28  2.46  2.63  2.82  3.03 ...
          3.25  3.49  3.74  4.02  4.32  4.64  4.98  5.35 ...
          6.15  7.07  8.10  9.25 10.54 11.97 13.56 15.31 ...
         17.23 19.34 21.64 24.15 26.88 29.84 33.05 36.52 ...
         40.25 44.27 48.59 53.22 58.18 63.49 68.00 68.00 ...
         68.00 68.00 68.00 68.00 68.00 68.00 68.00 68.00 ...
         68.00 68.00 68.00 68.00];

% Absolute hearing threshold as a function of the "k" indices
 LTq_k = zeros(250, 1);
 for i = 1:length(LTq_i)
    LTq_k(i_to_k(i)) = LTq_i(i);
 end
 for k = 1:250
    if LTq_k(k) == 0
        LTq_k(k) = last_nonzero;
    else
        last_nonzero = LTq_k(k);
    end
 end

 % Bark frequencies as a function of the "i" indices
 Table_z = [ .850 1.694 2.525 3.337 4.124 4.882 5.608 6.301 ...
            6.959  7.581  8.169  8.723  9.244  9.734 10.195 10.629 ...
           11.037 11.421 11.783 12.125 12.448 12.753 13.042 13.317 ...
           13.578 13.826 14.062 14.288 14.504 14.711 14.909 15.100 ...
           15.284 15.460 15.631 15.796 15.955 16.110 16.260 16.406 ...
           16.547 16.685 16.820 16.951 17.079 17.205 17.327 17.447 ...
           17.680 17.905 18.121 18.331 18.534 18.731 18.922 19.108 ...
           19.289 19.464 19.635 19.801 19.963 20.120 20.273 20.421 ...
           20.565 20.705 20.840 20.972 21.099 21.222 21.342 21.457 ...
           21.677 21.882 22.074 22.253 22.420 22.576 22.721 22.857 ...
           22.984 23.102 23.213 23.317 23.415 23.506 23.592 23.673 ...
           23.749 23.821 23.888 23.952 24.013 24.070 24.125 24.176 ...
           24.225 24.271 24.316 24.358 24.398 24.436 24.473 24.508 ...
           24.542 24.574 25 25];

 Frontieres_i = [1 2 3 5 6 8 9 11 13 15 17 20 23 27 32 37 ...
          45 50 55 61 68 75 81 93 106];

 Frontieres_k = zeros(1, length(Frontieres_i));
 for i = 1:length(Frontieres_i)
```

```
   Frontieres_k(i) = i_to_k(Frontieres_i(i));
end

% Determining the width of critical bands
f_250_c = [0 Frontieres_k 296];
f_250_d = [0 Frontieres_k 256];

upper_bound = 1;
lower_bound = 1;
while upper_bound < 250
   lower_bound = lower_bound + 1;
   for k = 1:25
       if lower_bound >= f_250_c(k) & lower_bound < f_250_c(k+1)
           width_low = f_250_c(k+1) - f_250_c(k);
           no_sam_low = f_250_c(k+1) - lower_bound;
       end
   end
   if no_sam_low >= ceil(width_low/2)
       width_frm = ceil(width_low/2);
   else
       for k = 1:25
           if upper_bound >= f_250_c(k) & upper_bound < f_250_c(k+1)
               width_high = f_250_c(k+1) - f_250_c(k);
               no_sam_high = upper_bound - f_250_c(k);
           end
       end
       no_sam_tot = no_sam_high + no_sam_low;
       width_frm = ceil((width_low*no_sam_low/no_sam_tot + ...
           width_high*no_sam_high/no_sam_tot)/2);
   end
   upper_bound = lower_bound + width_frm;
   Larg_f(lower_bound) = width_frm;
end

function [thresh1_db, thresh2_db, SMR_db_sb] = mpa1_mpeg(xn_mpa)

% Calculating the masking threshold for mpeg psychoacoustic model 1
% (cf pages 122-128 of the standard ISO/CEI 11172-3:1993 (F))
% Inputs
%    xn_mpa(512,1) : unweighted signal
%      and some constants defined in init_mpa1_mpeg.m
% Outputs
%    thresh1_db(1,256) : masking threshold in dB
%    thresh2_db(1,256) : min of previous in each of the sub-bands
%      true threshold used by the mpeg coder to realize the binary
%      allocation
%    SMR_db_sb(1,32)  : 27 signal-to-mask ratios,
%      the others are not used
%

global Fe LTq_i LTq_k Table_z Frontieres_i Frontieres_k Larg_f

% Stage 1 : FFT Analysis
% **********************
N = length(xn_mpa);
hann = sqrt(8/3)/2*[ones(N, 1) - cos(2*pi*(0:N-1)'/N)];
if sum(abs(xn_mpa)) > 0
   X1 = fft(xn_mpa.*hann);
   X1 = (abs(X1(1:N/2+1)).^2)/N;
   perio_xn_db = 10*log10(X1);
else
   perio_xn_db = zeros(N/2+1,1);
end
offset = max(perio_xn_db) - 96;
X = perio_xn_db - offset;


% Stage 2 : Determing the acoustic pressure level
% ***********************************************
% Not programmed

% Stage 3 : Taking the absolute hearing threshold into account
% ************************************************************
% Not programmed

% Stage 4 : Determining tonal and non-tonal components
% ****************************************************

% Finding local maxima
max_local = zeros(250, 1);
for k = 3:250
   if X(k) > X(k-1) & X(k) >= X(k+1)
       max_local(k) = 1;
   end
end

tonal = zeros(250, 1);
for k = 3:62
   if max_local(k)
       tonal(k) = 1;
       for j = [-2 2]
           if X(k) - X(k+j) < 7
               tonal(k) = 0;
           end
```

```
        end
    end
end

for k = 63:126
    if max_local(k)
        tonal(k) = 1;
        for j = [-3 -2 2 3]
            if X(k) - X(k+j) < 7
                tonal(k) = 0;
            end
        end
    end
end

for k = 127:250
    if max_local(k)
        tonal(k) = 1;
        for j = [-6:-2 2:6]
            if X(k) - X(k+j) < 7
                tonal(k) = 0;
            end
        end
    end
end

% Finding tonal divisions
X_tm = zeros(250,1);
for k = 1:250
    if tonal(k)
        time = 10^(X(k-1)/10) + 10^(X(k)/10) + 10^(X(k+1)/10);
        X_tm(k) = 10*log10(time);
        X(k-1) = -100;
        X(k)   = -100;
        X(k+1) = -100;
    else
        X_tm(k) = -100;
    end
end

X_nm = -100*ones(250, 1);
k = 1;
for k1 = Frontieres_k
    geom_mean = 1;
    pow = 0;
    raies_en_sb = 0;
```

```
   while k <= k1
       geom_mean = geom_mean*k;
       pow = pow + 10^(X(k)/10);
       k = k + 1;
       raies_en_sb = raies_en_sb + 1;
   end
   geom_mean = floor(geom_mean^(1/raies_en_sb));
   X_nm(geom_mean) = 10*log10(pow);
end

X_tm_avant = X_tm;
X_nm_avant = X_nm;

% Stage 5 : Removing masking components
% ************************************

for k = 1:250
   if X_tm(k) < LTq_k(k)
       X_tm(k) = -100;
   end
   if X_nm(k) < LTq_k(k)
       X_nm(k) = -100;
   end
end

% Eliminating neighboring tonal divisions: sliding frames
upper_bound = 1;
lower_bound = 1;
while upper_bound < 250
   [ans, max_ix] = max(X_tm(lower_bound:upper_bound));
   for k = lower_bound:upper_bound
       if k-lower_bound+1 ~= max_ix
           X_tm(k) = -100;
       end
   end
   lower_bound = lower_bound + 1;
   upper_bound = lower_bound + Larg_f(lower_bound);
end

% Stage 6: Calculating individual masking thresholds
% **************************************************

% Removing: "k" indices toward "i" indices
Num_comp_i = length(Table_z);
X_tm_i = -100*ones(Num_comp_i, 1);
X_nm_i = -100*ones(Num_comp_i, 1);
```

```
for k = 1:250
   if X_tm(k) >= -10
       X_tm_i(ppv(k)) = X_tm(k);
   end
end

for k = 1:250
   if X_nm(k) >= -10
       X_nm_i(ppv(k)) = X_nm(k);
   end
end

% Stage 6->7 : Calculating global masking thresholds
% **************************************************

thresh_m = zeros(Num_comp_i, 1);
no_tm = 0;
no_nm = 0;
for i = 1:Num_comp_i
   if X_tm_i(i) > -100
       no_tm = no_tm + 1;
   end
   if X_nm_i(i) > -100
       no_nm = no_nm + 1;
   end
end

tab_tm = zeros(1,no_tm);
tab_nm = zeros(1,no_nm);
ix = 1;
for i = 1:Num_comp_i
   if X_tm_i(i) > -100
       tab_tm(ix) = i;
       ix = ix + 1;
   end
end

ix = 1;
for i = 1:Num_comp_i
   if X_nm_i(i) > -100
       tab_nm(ix) = i;
       ix = ix + 1;
   end
end

for i = 1:Num_comp_i
   sum_tm = 0;
   z_i = Table_z(i);
   for j = tab_tm
       z_j = Table_z(j);
       dz = z_i - z_j;
       if dz >= -3 & dz < 8
           LT_tm = X_tm_i(j) + (-1.525 - 0.275*z_j - 4.5) + ...
               vf(dz, j, X_tm_i);
           sum_tm = sum_tm + 10 ^ (LT_tm/10);
       end
   end
   sum_nm = 0;
   for j = tab_nm
       z_j = Table_z(j);
       dz = z_i - z_j;
       if dz >= -3 & dz < 8
           LT_nm = X_nm_i(j) + (-1.525 - 0.175*z_j - 0.5) + ...
               vf(dz, j, X_nm_i);
           sum_nm = sum_nm + 10 ^ (LT_nm/10);
       end
   end
   thresh_m(i) = 10 * log10(10^(LTq_i(i)/10) + sum_tm + sum_nm);
end

% Stage 8: Determining
%           - the masking threshold
%           - the minimum masking threshold in each sub-band
%           - the signal-to-mask ratio
% ****************************************************************

thresh1_db = zeros(1,256);
thresh2_db = zeros(1,256);
for i = 1:6
   t1 = thresh_m(8*(i-1)+1:8*(i));
   thresh1_db(8*(i-1)+1:8*(i)) = t1;
   thresh2_db(8*(i-1)+1:8*(i)) = ones(1,8)*min(t1);
end
for i = 7:12
   i1 = i - 6;
   t1 = thresh_m(49+4*(i1-1):48+4*(i1));
   t2(1:2:7) = t1;
   t2(2:2:8) = t1;
   thresh1_db(8*(i-1)+1:8*(i)) = t2;
   thresh2_db(8*(i-1)+1:8*(i)) = ones(1,8)*min(t1);
end
for i = 13:30
   i1 = i - 12;
   t1 = thresh_m(73+2*(i1-1):72+2*(i1));
   t2(1:4:5) = t1;
   t2(2:4:6) = t1;
   t2(3:4:7) = t1;
   t2(4:4:8) = t1;
   thresh1_db(8*(i-1)+1:8*(i)) = t2;
   thresh2_db(8*(i-1)+1:8*(i)) = ones(1,8)*min(t1);
end
for i = 31:32
   thresh1_db(8*(i-1)+1:8*(i)) = ones(1,8)*min(t1);
   thresh2_db(8*(i-1)+1:8*(i)) = ones(1,8)*min(t1);
end

thresh1_db = thresh1_db + offset;
thresh2_db = thresh2_db + offset;
SMR_db_sb = zeros(1,32);
for i = 1:32
   SMR_db_sb(i) = max(perio_xn_db((i-1)*8+1:i*8)) - thresh2_db(i*8);
end

function i0 = ppv(k0)
if k0 <= 48
   i0 = k0;
elseif k0 <= 96
   i0 = floor((k0-48)/2) + 48;
else
   i0 = round((k0-96)/4) + 72;
end
if i0 > 108
   i0 = 108;
end

function le_vf = vf(dz, j, X)
if dz < -1
   le_vf = 17 * (dz + 1) - (0.4 * X(j) + 6);
elseif dz < 0
   le_vf = (0.4 * X(j) + 6) * dz;
elseif dz < 1
   le_vf = -17 * dz;
else
   le_vf = -(dz - 1) * (17 - 0.15 * X(j)) - 17;
end

function NOQ = mpeg_alloc(SMR, Bitrate_kbs)

% Procedure for allocating bits

global Fe
global Rsb_iso_1_3 Rsb_iso_4_11 Rsb_iso_12_23 Rsb_iso_24_27

SMR = [SMR(1:27) zeros(1,5)];
Bits_disp_init = Bitrate_kbs*1000*384/Fe - 88;
Bits_disp = Bits_disp_init;
NOQ = ones(1,27);
NOQ_max = [15*ones(1,11) 7*ones(1,12) 3*ones(1,4)];
MNR = -SMR(1:27);
MNR_init = MNR;
SNR = zeros(1,27);
Num_bits = zeros(1,27);
Scf = zeros(1,27);

iter = 0;
iter_max = 1000;
while Bits_disp > 0
   [time kmin] = min(MNR);
   if NOQ(kmin) == NOQ_max(kmin)
       MNR(kmin) = 100;
   else
       Scf(kmin) = 1;
       NOQ(kmin) = NOQ(kmin) + 1;
       if kmin < 4
           SNR(kmin)       = Rsb_iso_1_3(NOQ(kmin),2);
           Num_bits(kmin) = Rsb_iso_1_3(NOQ(kmin),3);
       elseif kmin < 12
           SNR(kmin)       = Rsb_iso_4_11(NOQ(kmin),2);
           Num_bits(kmin) = Rsb_iso_4_11(NOQ(kmin),3);
       elseif kmin < 24
           SNR(kmin)       = Rsb_iso_12_23(NOQ(kmin),2);
           Num_bits(kmin) = Rsb_iso_12_23(NOQ(kmin),3);
       elseif kmin < 28
           SNR(kmin)       = Rsb_iso_24_27(NOQ(kmin),2);
           Num_bits(kmin) = Rsb_iso_24_27(NOQ(kmin),3);
       end
       MNR(kmin) = SNR(kmin) - SMR(kmin);
       Bits_disp_old = Bits_disp;
       Bits_disp = Bits_disp_init - 6*sum(Scf) - 12*sum(Num_bits);
   end
   iter = iter + 1;
   if iter > iter_max
       fprintf('Bits_disp_init = %d Bits_disp = %d \n', ...
           Bits_disp_init, Bits_disp)
       return
   end
end
```

```
function write_bit_stream(fid_bit_stream, NOQ, Mat_sb)

% Quantizing the sub-band signals matrix and constructing
% the bit stream

global Scl_fact Rsb_iso_1_3 Rsb_iso_4_11 Rsb_iso_12_23 Rsb_iso_24_27

% Firstly, we write the binary allocation
for k = 1:11
   fwrite(fid_bit_stream, NOQ(k)-1, 'ubit4');
end
for k = 12:23
   fwrite(fid_bit_stream, NOQ(k)-1, 'ubit3');
end
for k = 24:27
   fwrite(fid_bit_stream, NOQ(k)-1, 'ubit2');
end
for k = 1:27
   if k < 4
       Num_pas_quantif(k) = Rsb_iso_1_3(NOQ(k),1);
   elseif k < 12
       Num_pas_quantif(k) = Rsb_iso_4_11(NOQ(k),1);
   elseif k < 24
       Num_pas_quantif(k) = Rsb_iso_12_23(NOQ(k),1);
   elseif k < 28
       Num_pas_quantif(k) = Rsb_iso_24_27(NOQ(k),1);
   end
end

% Next, we write the scale factors
for k = 1:27
   if Num_pas_quantif(k) > 0
       vmax = max(abs(Mat_sb(:,k)));
       indice_Scf = 63;
       while vmax > Scl_fact(indice_Scf) & indice_Scf >= 2
           indice_Scf = indice_Scf - 1;
       end
       if vmax > Scl_fact(1)
           indice_Scf = 1;
       end
       fwrite(fid_bit_stream, indice_Scf, 'ubit6');
       Scf(k) = Scl_fact(indice_Scf);
   end
end

% Finally, we write the sub-band samples
for k = 1:27
```

```
   if Num_pas_quantif(k) > 0
       alpha = 2*Scf(k)/Num_pas_quantif(k);
       dico = -Scf(k)+alpha/2:alpha:Scf(k)-alpha/2;
       indices_ech = zeros(12, 1);
       for n = 1:12
           [val indice] = min(abs(dico - Mat_sb(n,k)));
           indices_ech(n) = indice;
       end

       if Num_pas_quantif(k) <= 9
           num_bits_prec = ceil(log2(Num_pas_quantif(k)^3));
           precision = ['ubit' int2str(num_bits_prec)];
           carre = Num_pas_quantif(k)^2;
           for n = 1:4
               n1 = (n-1)*3 + 1;
               ind = indices_ech(n1)-1 + ...
                   (indices_ech(n1+1)-1)*Num_pas_quantif(k) + ...
                   (indices_ech(n1+2)-1)*carre;
               fwrite(fid_bit_stream, ind, precision);
           end
       else
           num_bits_prec = log2(Num_pas_quantif(k)+1);
           precision = ['ubit' int2str(num_bits_prec)];
           for n = 1:12
               fwrite(fid_bit_stream, indices_ech(n)-1, precision);
           end
       end
   end
end

function [Mat_sb_q, stop] = lit_bit_stream(fid_bit_stream)

% Reading the bitstream
% reconstructing the sub-band signals matrix

global Scl_fact Rsb_iso_1_3 Rsb_iso_4_11 Rsb_iso_12_23 Rsb_iso_24_27

Mat_sb_q = zeros(12,32);
stop = 0;

% Firstly, we read the binary allocation
for k = 1:11
   [val num] = fread(fid_bit_stream, 1, 'ubit4');
   if num == 0
       stop = 1;
       return
   end
   NOQ(k) = val + 1;
   if k < 4
       Num_pas_quantif(k) = Rsb_iso_1_3(NOQ(k),1);
   elseif k < 12
       Num_pas_quantif(k) = Rsb_iso_4_11(NOQ(k),1);
   end
end
for k = 12:23
   [val num] = fread(fid_bit_stream, 1, 'ubit3');
   if num == 0
       stop = 1;
       return
   end
   NOQ(k) = val + 1;
   Num_pas_quantif(k) = Rsb_iso_12_23(NOQ(k),1);
end
for k = 24:27
   [val num] = fread(fid_bit_stream, 1, 'ubit2');
   if num == 0
       stop = 1;
       return
   end
   NOQ(k) = val + 1;
   Num_pas_quantif(k) = Rsb_iso_24_27(NOQ(k),1);
end

% Next, scale factors
for k = 1:27
   if Num_pas_quantif(k) > 0
       indice_Scf = fread(fid_bit_stream, 1, 'ubit6');
       Scf(k) = Scl_fact(indice_Scf);
   end
end

% Finally, the sub-band samples
for k = 1:27
   if Num_pas_quantif(k) > 0
       alpha = 2*Scf(k)/Num_pas_quantif(k);
       dico = -Scf(k)+alpha/2:alpha:Scf(k)-alpha/2;
       if Num_pas_quantif(k) <= 9
           num_bits_prec = ceil(log2(Num_pas_quantif(k)^3));
           precision = ['ubit' int2str(num_bits_prec)];
           carre = Num_pas_quantif(k)^2;
           for n = 1:4
               ind = fread(fid_bit_stream, 1, precision);
               n1 = (n-1)*3 + 1;
               ind3 = fix(ind/carre);
```

```
                ind2 = fix((ind - carre*ind3)/Num_pas_quantif(k));
                ind1 = ind - carre*ind3 - Num_pas_quantif(k)*ind2;
                Mat_sb_q(n1,k) = dico(ind1+1);
                Mat_sb_q(n1+1,k) = dico(ind2+1);
                Mat_sb_q(n1+2,k) = dico(ind3+1);
            end
        else
            num_bits_prec = log2(Num_pas_quantif(k)+1);
            precision = ['ubit' int2str(num_bits_prec)];
            for n = 1:12
                indice_ech = fread(fid_bit_stream, 1, precision);
                Mat_sb_q(n,k) = dico(indice_ech+1);
            end
        end
    else
        Scf(k) = 0;
        Mat_sb_q(:,k) = zeros(12,1);
    end
end
```

Bibliography

[ADV 95] Advanced Television Systems Comittee, Digital audio compression standard (AC-3), p. 1–140, 1995.

[ATA 79] ATAL B., SCHROEDER M., "Predictive coding of speech signals and subjective error criteria", *IEEE Transactions on Acoustics, Speech, and Signal Processing*, vol. ASSP-27, p. 247–254, 1979.

[ATA 82] ATAL B., REMDE J., "A new model of LPC excitation for producing natural-sounding speech at low bit rates", *Proceedings of the International Conference on Acoustics, Speech, and Signal Processing*, vol. 1, p. 614–617, 1982.

[BAU 03] BAUMGARTE F., FALLER C., "Binaural cue coding – Part I: Psychoacoustic fundamentals and design principles", *IEEE Transactions on Speech and Audio Processing*, vol. 11, no. 6, p. 509–519, 2003.

[BER 71] BERGER T., *Rate-distortion Theory: A Mathematical Basis for Data Compression*, Prentice-Hall, 1971.

[BES 02] BESSETTE B., SALAMI R., LEFEBVRE R., JELINEK M., ROTOLA-PUKKILA J., VAINIO J., MIKKOLA H., JARVINEN K., "The adaptive multirate wideband speech codec (AMR-WB)", *IEEE Transactions on Speech and Audio Processing*, vol. 10, no. 8, p. 620–636, 2002.

[BLA 87] BLAHUT R., *Principles and Practice of Information Theory*, Addison-Wesley, 1987.

[BRE 05] BREEBAART J., VAN DE PAR S., KOHLRAUSCH A., SCHUIJERS E., "Parametric coding of stereo audio", *EURASIP Journal on Applied Signal Processing*, vol. 9, p. 1305–1322, 2005.

[BRI 07] BRIAND M., VIRETTE D., MARTIN N., "Parametric representation of multichannel audio based on principal component analysis", *120th Convention AES*, May 2006.

[CAL 02] CALVET O., *Acoustique appliquée aux techniques du son*, Editions Casteilla, 2002.

[CCI 80] International Telecommunication Union (ITU-T), Recommendation T.4: Procedures for document facsimile transmission in the general switched telephone network, 1980.

[COL 02] DIETZ M., LILJERYD L., KJORLING K., KUNZ, "Spectral Band Replication, a novel approach in audio coding", *120th Convention AES*, May 2002.

[COV 91] COVER T., THOMAS J., *Elements of Information Theory*, Wiley, Series in Telecommunications, 1991.

[DER 00] DERRIEN O., LARBI S., PERREAU-GUIMARAES M., MOREAU N., “Le codeur MPEG-2 AAC expliqué aux traiteurs de signaux”, *Annales des Télécommunications*, September–October 2000.

[EUR 96] European Telecommunication Standard, Digital cellular telecommunications system; Enhanced Full Rate (EFR) speech transcoding (GSM 06.60), Final Draft 1996.

[FAL 03] FALLER C., BAUMGARTE F., “Binaural cue coding – Part II: Schemes and Applications”, *IEEE Transactions on Speech and Audio Processing*, vol. 11, no. 6, p. 520–531, 2003.

[FAL 04] FALLER C., “Parametric coding of spatial audio”, PhD thesis, Ecole Polytechnique Fédérale de Lausanne, 2004.

[GAL 68] GALLAGER R., *Information Theory and Reliable Communication*, Wiley, 1968.

[GEI 06] GEIGER R., YU R., HERRE J., RAHARDJA S., KIM S., LIN X., SCHMIDT M., “ISO/IEC MPEG-4 high-definition scalable advanced audio coding”, *120th Convention AES*, May 2006.

[GER 92] GERSHO A., GRAY R., *Vector Quantization and Signal Compression*, Kluwer Academic Publishers, 1992.

[GOL 89] GOLUB G., VAN LOAN C., *Matrix Computations*, Johns Hopkins University Press, 1983 (Second Edition 1989).

[GRA 72] GRAY R., “On the asymptotic eigenvalue distribution of Toeplitz matrices”, *IEEE Transactions on Information Theory*, vol. IT-18, no. 6, p. 725–730, 1972.

[GRA 90] GRAY R., *Source Coding Theory*, Kluwer Academic Publishers, 1990.

[HAN 01] HANS M., SCHAFER R., “Lossless compression of digital audio”, *IEEE Signal Processing Magazine*, p. 21–32, July 2001.

[HAY 91] HAYKIN S., *Adaptive Filter Theory*, Prentice Hall, 1991 (2nd edition).

[HER 04] HERRE J., “From joint stereo to spatial audio coding – Recent progress and standardization”, *Proceedings of the 7th International Conference on Digital Audio Effects*, 2004.

[HOW 94] HOWARD P., VITTER J., “Arithmetic coding for data compression”, *Processing of the IEEE*, vol. 82, no. 6, p. 857–865, 1994.

[INT 97] International Organization for Standardization, ISO/IEC 13818-7 (MPEG-2 Advanced Audio Coding, AAC), 1997.

[INT 98] International Organization for Standardization, ISO/IEC 14496-3 (Information technology - Very low bitrate audio-visual coding), 1998.

[INT 05] International Organization for Standardization, ISO/IEC 14496-3:2005 (Information technology - Coding of audio-visual objects), 2005.

[ITA 75] ITAKURA F., “Line spectrum representation of linear predictive coefficients of speech signals”, *Journal of the Acoustical Society of America*, vol. 57, 1975.

[JAY 84] JAYANT N., NOLL P., *Digital Coding of Waveforms*, Prentice Hall, 1984.

[KAY 88] KAY S., *Modern Spectral Estimation – Theory and Application*, Prentice Hall, 1988.

[KON 03] KONSTANTINIDES K., "An introduction to Super Audio CD and DVD-Audio", *IEEE Signal Processing Magazine*, July 2003.

[LAP 06] LAPIERRE J., LEFEBVRE R., "On improving parametric stereo audio coding", *120th Convention AES*, May 2006.

[LIN 80] LINDE Y., BUZO A., GRAY R., "An algorithm for vector quantizer design", *IEEE Transactions on Communications*, vol. COM-28, p. 84–95, 1980.

[LIP 92] LIPSHITZ S., WANNAMAKER R., VANDERKOOY J., "Quantization and dither: a theoretical survey", *Journal of the Audio Engineering Society*, vol. 40, no. 5, p. 355–375, 1992.

[MAL 92] MALVAR H., *Signal Processing with Lapped Transforms*, Artech House, 1992.

[MOO 82] MOORE B., *An Introduction to the Psychology of Hearing*, Academic Press, Second edition, 1982.

[NOR 93] International Standard ISO/IEC 11172-3, Coding of moving pictures and associated audio for digital storage media at up to about 1.5 Mbit/s, 1993.

[PAI 00] PAINTER T., SPANIAS A., "Perceptual coding of digital audio", *Proceedings of the IEEE*, vol. 88, p. 451–513, 2000.

[PUR 04] PURNHAGEN H., "Low complexity parametric stereo coding in MPEG-4", *Proceedings of the 7th International Conference on Digital Audio Effects*, October 2004.

[RIO 93] RIOUL O., VETTERLI M., "Wavelets and signal processing", *IEEE Signal Processing Mag.*, vol. 8, p. 14–38, 1991.

[SAL 98] SALAMI R., LAFLAMME C., ADOUL J., KATAOKA A., HAYASHI S., LAMBLIN C., MASSALOUX D., PROUST S., KROON P., SHOHAM Y., "Design and description of CS-ACELP: a toll quality 8 kb/s speech coder", *IEEE Transactions on Speech and Audio Processing*, vol. 6, no. 2, p. 116–130, 1998.

[SCH 85] SCHROEDER M., ATAL B., "Code-excited linear prediction (CELP): high-quality speech at very low bit rates", *Proceedings on the International Conference on Acoustics, Speech, and Signal Processing*, p. 937–940, 1985.

[SHA 48] SHANNON C., "A mathematical theory of communication", *Bell System Technical Journal*, vol. 27, p. 379–423 and 623–656, 1948.

[SHA 59] SHANNON C., "Coding theorems for a discrete source with a fidelity criterion", *IRE National Convention Record*, vol. 4, p. 142–163, 1959.

[SHO 88] SHOHAM Y., GERSHO A., "Efficient bit allocation for an arbitrary set of quantizers", *IEEE Transactions on Acoustics, Speech, and Signal Processing*, vol. 36, no. 9, p. 1445–1453, 1988.

[SLO 84] SLOANE N., "The packing of spheres", *Scientific American*, p. 116–125, January 1984.

[UNI 96] International Telecommunication Union (ITU-T), Recommendation G.729: Coding of speech at 8 kbit/s using conjugate-structure algebraic-code-excited linear prediction (CS-ACELP), 1996.

[UNI 01] International Telecommunication Union (ITU-R), Recommendation BS.1387: Method for objective measurements of perceived audio quality, 2001.

[VAI 90] VAIDYANATHAN P., "Multirate digital filters, filter banks, polyphase networks and applications: A tutorial", *Proceedings of the IEEE*, January 1990.

[YU 06] YU R., RAHARDJA S., XIAO L., KO C., "A fine granular scalable to lossless audio coder", *IEEE Transactions on Audio, Speech, and Language Processing*, vol. 14, p. 1352–1363, 2006.

[ZAD 82] ZADOR P., "Asymptotic quantization error of continuous signals and the quantization dimension", *IEEE Transactions on Information Theory,* vol. IT-28, p. 139–149, 1982.

[ZWI 81] ZWICKER E., FELDTKELLER E., *Das ohr als nachrichtenempfanger*, S. Hirzel Verlag, Stuttgart, 1956.

Index