

1-1-1996

A critical evaluation of the Dolby Digital compression algorithm

David Andrew Ingbretsen
University of Nevada, Las Vegas

Follow this and additional works at: <https://digitalscholarship.unlv.edu/rtds>

Repository Citation

Ingbretsen, David Andrew, "A critical evaluation of the Dolby Digital compression algorithm" (1996). *UNLV Retrospective Theses & Dissertations*. 3279.

<http://dx.doi.org/10.25669/dw8q-xx1b> processed, response: 201

This Thesis is protected by copyright and/or related rights. It has been brought to you by Digital Scholarship@UNLV with permission from the rights-holder(s). You are free to use this Thesis in any way that is permitted by the copyright and related rights legislation that applies to your use. For other uses you need to obtain permission from the rights-holder(s) directly, unless additional rights are indicated by a Creative Commons license in the record and/or on the work itself.

This Thesis has been accepted for inclusion in UNLV Retrospective Theses & Dissertations by an authorized administrator of Digital Scholarship@UNLV. For more information, please contact digitalscholarship@unlv.edu.

INFORMATION TO USERS

This manuscript has been reproduced from the microfilm master. UMI films the text directly from the original or copy submitted. Thus, some thesis and dissertation copies are in typewriter face, while others may be from any type of computer printer.

The quality of this reproduction is dependent upon the quality of the copy submitted. Broken or indistinct print, colored or poor quality illustrations and photographs, print bleedthrough, substandard margins, and improper alignment can adversely affect reproduction.

In the unlikely event that the author did not send UMI a complete manuscript and there are missing pages, these will be noted. Also, if unauthorized copyright material had to be removed, a note will indicate the deletion.

Oversize materials (e.g., maps, drawings, charts) are reproduced by sectioning the original, beginning at the upper left-hand corner and continuing from left to right in equal sections with small overlaps. Each original is also photographed in one exposure and is included in reduced form at the back of the book.

Photographs included in the original manuscript have been reproduced xerographically in this copy. Higher quality 6" x 9" black and white photographic prints are available for any photographs or illustrations appearing in this copy for an additional charge. Contact UMI directly to order.

UMI

A Bell & Howell Information Company
300 North Zeeb Road, Ann Arbor MI 48106-1346 USA
313/761-4700 800/521-0600

**A CRITICAL EVALUATION OF THE DOLBY DIGITAL
COMPRESSION ALGORITHM**

by

David A. Ingbretsen

**A thesis submitted in partial fulfillment
of the requirements of the degree of**

Master of Science

in

Computer Science

**Department of Computer Science
University of Nevada, Las Vegas
May 1997**

UMI Number: 1385177

UMI Microform 1385177
Copyright 1997, by UMI Company. All rights reserved.

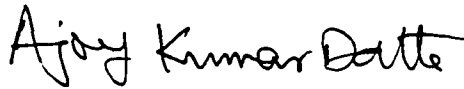
**This microform edition is protected against unauthorized
copying under Title 17, United States Code.**

UMI
300 North Zeeb Road
Ann Arbor, MI 48103


The Thesis of David A. Ingbretsen for the degree of Master of Science
in Computer Science is approved.



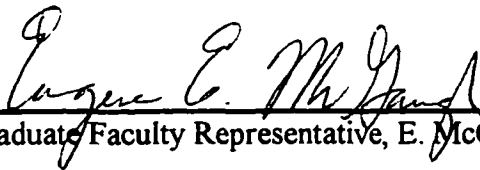
Chairperson, E. A. Yfantis, Ph.D.



Examining Committee Member, A. Datta, Ph.D.



Examining Committee Member, L. Gewali, Ph.D.



Graduate Faculty Representative, E. McGaugh, Ph.D.



Dean of the Graduate College, R. Smith, Ph.D.

University of Nevada, Las Vegas
May 1997

ABSTRACT

Dolby Digital, or AC-3, provides for the compression of high-quality audio signals at a bit rate of 384 kbps. for 5.1 audio channels. The purpose of this thesis is to conduct an analysis and evaluation of the AC-3 compression algorithm. The main conclusion is that AC-3 allocates too few bits to the upper frequency ranges. This conclusion is supported by two lines of argument. First, a comparison of the AC-3 perceptual coding model with known facts concerning the masking properties of sound reveals omissions in the AC-3 perceptual model that lead to a loss of high frequency resolution. Second, a test program comparing the bit allocation generated by AC-3 with the bit allocation generated by MPEG corroborates the conclusion. A remedy for the high frequency loss problem is put forward.

TABLE OF CONTENTS

ABSTRACT	iii
LIST OF TABLES	vi
ACKNOWLEDGEMENTS	vii
CHAPTER 1 INTRODUCTION TO AUDIO COMPRESSION	1
I. Overview	1
II. The Physics of Sound	2
A. The nature of waves	2
B. Mathematical properties of sound waves	3
C. The measurement of sound	5
III. Digitized Sound	5
A. Digital sampling	5
B. The Nyquist sampling rate	6
IV. Quantization	7
V. Perceptual Coding	9
A. Basic concepts	9
B. Threshold in quiet	10
C. Critical bands	11
VI. Criteria for Evaluating a Perceptual Coder	13
CHAPTER 2 THE MODIFIED DISCRETE COSINE TRANSFORM	15
I. Introduction	15
II. The Forward and Inverse Fourier Transforms	17
III. The Forward and Inverse Discrete Cosine Transforms	19
IV. The Windowing Function	22
V. A Generalized Windowing Function	25
CHAPTER 3 THE AC-3 CODEC	32
I. Introduction	32
II. Overview of AC-3	33
III. Representation of Exponents	35

CHAPTER 4 PERCEPTUAL CODING	39
I. Introduction	39
II. Types of Masking Effects	40
A. Masking effects of noise	41
B. Masking effects of tones	43
C. The spreading function	44
D. Summary of masking effects	45
III. Summary of the AC-3 Masking Algorithm	46
A. Power spectral density function	47
B. The excitation function	47
C. Bit allocation	51
IV. Assessment of the AC-3 Masking Algorithm	52
A. Intra-band tonal and non-tonal masking	53
B. Location within a band	56
C. Masking at higher bark levels	56
D. Spreading function: upper and lower slopes	57
E. Spreading function and bark value	58
F. Spreading function and amplitude	58
G. Conclusions	58
V. The MPEG Masking Algorithm	59
A. Calculation of the power spectral density function	59
B. Identification of tonal and non-tonal maskers	60
C. Calculation of individual masking thresholds	61
D. The global masking calculation	64
VI. Assessment of the MPEG masking algorithm	64
 CHAPTER 5 EVALUATION OF THE AC-3 CODER	 67
I. Unsuccessful Revision Strategies	67
A. The exponent strategies	67
B. The masking algorithm	72
II. Underquantization of High Frequencies	74
III. Modifications to the Default Bit Allocation	77
IV. A Remedy for High Frequency Loss.....	79
 BIBLIOGRAPHY	 83

LIST OF TABLES

Table 1 Standard Audio Types.....	6
Table 2 Critical Bands.....	12
Table 3 Bit Allocation (csnroffst = 10).....	55
Table 4 Bit Allocation (csnroffst=15).....	55
Table 5 Masking Index Values.....	62
Table 6 Spreading Function Values.....	63
Table 7 Exponent Mean Squared Error.....	69
Table 8 Exponent Strategies and Bit Allocation.....	70
Table 9 AC-3 Bit Allocation, Jazz Selection.....	76
Table 10 AC-3 Bit Allocation, Scraping Metals.....	76
Table 11 Modified Bit Allocation: Jazz Selection.....	81
Table 12 Modified Bit Allocation: Scraping Metals.....	82

ACKNOWLEDGEMENTS

I wish to thank Prof. E. Yfantis for his assistance. In particular, Dr. Yfantis gave generously of his time and greatly improved my understanding of the material in Chapter Two.

CHAPTER 1

INTRODUCTION TO AUDIO COMPRESSION

I. Overview

Dolby Digital, also known as AC-3, is one of the most important audio compression algorithms currently available. It has been accepted as the audio compression standard for high definition television in the United States, and will be at least one of the audio compression schemes available on digital video disks. It supports the encoding of 5.1 audio channels at a bit rate of 384 kbps, and is primarily designed to encode compact-disk-quality audio signals rather than radio signals or speech. AC-3 employs a technique known as 'perceptual coding'. A perceptual coder works by first taking the spectrum of the sound signal by means of a Fourier transform. It then makes use of principles of acoustics to distinguish portions of the spectrum that are audible from those that are masked by other portions of the spectrum. It is this acoustic analysis that is primarily responsible for the compression of the sound file.

The purpose of this essay is to describe and evaluate the AC-3 compression algorithm. The remainder of the present chapter provides background information on sound and audio compression. Chapter Two gives a detailed account of the particular version of the Fourier transform used by AC-3—the modified discrete cosine transform. Chapter Three describes the main features of the AC-3 compression algorithm. The fourth

and fifth chapters provide a critical assessment of AC-3. Chapter Four describes the known facts about the masking properties of sound. This description leads to the conclusion that the AC-3 perceptual coding algorithm does not take into account several important masking effects of sound. In Chapter Five various attempts to improve AC-3 are considered, and most are rejected as unproductive. However, it is shown, based on the discussion in the fourth chapter, that one consequence of the AC-3 perceptual coding algorithm is that too few bits are allocated to the higher frequencies. This conclusion is supported by the results of data derived from a test program that implements the AC-3 codec. A remedy for the high frequency loss problem is put forward and tested.

II. The Physics of Sound

A. The nature of waves

Physicists describe sound as a 'wave'—a periodic disturbance that moves through a medium.¹ A wave transfers energy over a distance without a corresponding movement of particles over that distance. When a wave moves through the ocean, for instance, individual particles of water do not actually travel along the wave, but are only displaced by a small amount. The displacement of some particles serves to transfer energy to neighboring particles, thereby advancing the movement of the wave. There are two general types of waves: transverse and longitudinal. In a transverse wave the displacement of particles occurs in a direction perpendicular to the direction of movement of the wave as a whole. A longitudinal wave, in contrast, displaces particles in the same direction as the movement of the wave. A wave in water is transverse; sound is a longitudinal wave.

As an example of a longitudinal wave, consider a metal spring stretched out parallel to the ground and secured at both ends. Suppose that, when viewed from left to right, P, Q and R are three consecutive sections of the spring. If Q is pulled in the direction of R and then released an 'area of compression' is created in the vicinity of QR in the sense that a greater portion of the spring is located in that area than elsewhere. A corresponding 'area of rarefaction' is created in the original location of Q. The higher density of the spring near QR creates pressure on Q to move back to its original location, and forces R to move further to the right. As R moves to the right additional areas of compression and rarefaction are generated, causing the wave to move along the spring. As Q moves back to the left its energy carries it beyond its original position, creating an area of compression with P, the segment of the spring originally to the left of Q. The result is that an oscillation is established. This effect, of course, is much more pronounced when sound moves through air than with a metal spring. If one imagines that air is trapped in a cylinder and pressure is applied at one end, it is easier to envision the pulsing oscillation of areas of compression and rarefaction that are created.

B. Mathematical properties of sound waves

The period (T) of a wave is the time required for one complete wave to pass through a point. The frequency (f) is the number of times per second that an individual particle in the wave completes one complete cycle or oscillation. It corresponds to the pitch or tone of a sound. There are two different measures of frequency. Stated in hertz, the frequency represents the number of cycles per second. Alternatively, the angular

¹ Much of the material in this section is based on Williams, Trinklein, Metcalfe & Lefler (1979, Chapters 10-11).

frequency (ω) measures the number of cycles completed over a 2π interval. The relationship between the two frequency measures is given by $\omega = 2\pi f$. The period T is the reciprocal of the frequency: if the frequency is 40 Hz., then $1/40$ of a second is required for one wave to pass through a point.

The wavelength (λ) is the distance between a point on one wave and the corresponding point on the next wave. The speed of the wave $v = \lambda/T$ because λ is the distance traveled, and T is the time required to travel one unit of distance. Since $f = 1/T$, it follows that $v = f\lambda$.

The amplitude (A) is the maximum displacement achieved by a particle in the wave. It roughly corresponds to the loudness of a sound. The phase (ϕ) represents a time shift: given some time that has been selected as t_0 , the phase can represent the wave at another time as a shifted replica of the wave at t_0 .

A sound wave can be represented by a sine curve, or a sum of sine curves. The sine function is a logical way to model a sound wave since both a wave and the sine function are periodic. The sine wave provides an effective mathematical model for a sound wave because three pieces of information are needed to fully represent the wave: its frequency (or angular frequency), its maximum amplitude, and its phase. The equation of a sine curve provides all three pieces of information:

$$x_a(t) = A \sin(2\pi ft + \phi) \text{ or}$$

$$x_a(t) = A \sin(\omega t + \phi).$$

In these equations ω and $2\pi f$ represent the frequency. When the sine function is graphed, A determines the maximum height of the curve (its amplitude), f (or ω) determines how

tight or spread out the curve appears (frequency), and ϕ determines how far the curve is shifted left or right along the x axis (the phase). Finally, $x_s(t)$ represents the momentary amplitude, or the amplitude at a given point in time.

C. The measurement of sound

The most important measure of the strength of an audio signal is the Sound Pressure Level. 'Pressure' is defined as the amount of force applied per unit of area, where force is measured in dynes and area in cm^2 (Luce, 1993, p. 44). The formula for the Sound Pressure Level is $\text{SPL} = 20 \log_{10} (P / P_0)$. P_0 is a reference level and is equal to 0.00002. Thus the Sound Pressure Level, in decibels, is $20 \log_{10} (P / 0.00002)$.

III. Digitized Sound

The process of digitizing sound modifies the analog sound wave in two respects: a continuous wave is replaced by samples of the wave taken at definite time intervals, and the indefinite precision of the analog wave gives way to a quantized representation of limited precision. A brief overview of each of these modifications is presented below.

A. Digital sampling

While sound waves could in theory be sampled at virtually any rate, in practice certain sampling rates have been adopted as recognized standards. Table 1 (adapted from Jayant & Chen, 1995, p, 24) summarizes the most important sampling rates.

Table 1 Standard Audio Types

Audio Grade	Bandwidth (Hz)
Telephone Speech	200-3200
Wideband Audio	50-7000
AM Audio	50-7000
FM Audio	50-15000
CD Audio	20-20000

Audible sounds range in frequency from 20-20000 Hertz. Human voices operate in the much more limited range of about 50-8000 Hertz. In this essay attention will be limited to the compression of sound waves that span the entire audible range from 20-20000 Hertz, which is the range spanned by high quality audio.

B. The Nyquist sampling rate

The 'sampling frequency' is the number of digital samples taken per second. When a sound file is digitized it is essential that the sampling frequency be sufficiently high to ensure an accurate representation of the waveform. According to the Nyquist sampling theorem, the sampling frequency must be at least twice the bandwidth of the signal in order to avoid aliasing (Proakis & Manolakis, 1992, p. 27). An example will illustrate the need for this restriction. Let F_s be the sampling frequency, F the frequency of the analog signal, and f the frequency of the digitized signal. Consider two analog signals:

$$x_1(t) = \cos(2\pi 10t);$$

$$x_2(t) = \cos(2\pi 50t).$$

Let us suppose a digital signal is generated using a sampling frequency of 40 Hz. Since $f = F/F_s$, it follows that the discrete versions of the two signals are given by:

$$x_1(n) = \cos(2\pi 10t/40) = \cos(\pi/2);$$

$$x_2(n) = \cos(2\pi 50 t/40) = \cos(5\pi/2).$$

Since $\cos(\pi/2) = \cos(5\pi/2)$, these two signals will produce overlapping sets of values.

There is no way to determine whether sampled values should be attributed to x_1 or to x_2 .²

On the other hand, if the sampling frequency is at least twice the largest frequency the problem of aliasing does not arise. In this example the largest frequency is 50, so the problem of aliasing is avoided by setting F_s to 120. On this interpretation the digital signals are given by:

$$x_1(n) = \cos(2\pi 10/120) = \cos(\pi/6);$$

$$x_2(n) = \cos(2\pi 50/120) = \cos(5\pi/6).$$

It is clear that the problem of aliasing does not arise provided that all the cosine values are less than or equal to π . The Nyquist sampling rate requirement ensures that the range between π and 2π is unused, thereby preventing aliasing.

IV. Quantization

Quantization is the process of replacing an analog version of the signal having potentially unlimited precision, with a digital representation of limited precision. The standard format for the quantization of sound is Pulse Code Modulation (PCM). There are several different versions of PCM. An early official version for the representation of speech is contained in International Telecommunication Union (1988). This specification

² This example is adapted from Proakis & Manolakis (1992, p. 23).

defines a method of encoding PCM using eight bits per sample at a sampling rate of 8000 samples per second, with the numbers converted to a logarithmic scale.

A more common version of PCM is the one used in .wav files. It uses a linear rather than a logarithmic scale, supports both 8-bit and 16-bit samples, and allows for virtually any sampling rate. The 8-bit format uses 256 numbers to represent the amplitudes in the sound file. The relation between the 8-bit quantized value and the actual amplitude depends on the sampling frequency. For example, if the sound wave is sampled at a rate of 8000 samples per second, then the amplitudes range from -4000 to +4000. The 8-bit values from 0 to 127 correspond to negative amplitudes, 128 corresponds to an amplitude of zero, and the numbers from 129 to 255 correspond to the positive amplitudes. A linear scale is used to map the 8-bit values onto the positive and negative amplitudes. The number 255 corresponds to the largest positive amplitude, and 0 corresponds to the smallest negative amplitude.

16-bit PCM employs a similar approach, using a linear scale to map the quantized values onto the amplitudes. The most important difference between the 8-bit and 16-bit formats is that the latter uses a much finer scale to quantize the amplitudes. Another difference is that 16-bit PCM uses the most significant bit to represent the sign. The quantized values from 0 to 32767 correspond to positive amplitudes, and the remaining values correspond to negative amplitudes. 16-bit PCM is the encoding method used on audio compact disks, and is thus the most important format for high quality audio.

V. Perceptual Coding

A. Basic concepts

Any reasonable form of lossy audio compression that strives for high compression rates must use some form of variable bit allocation. A Fourier transform is used to obtain the frequency coefficients that comprise the spectrum of the signal, and these coefficients are then transmitted in the compressed file. One cannot, however, simply allocate the same number of bits for the quantization of each coefficient. Current audio coders aim for an average of 1.5 to 3 bits per coefficient. Such a small number of bits would be incapable of providing adequate sound quality if evenly distributed among all coefficients. The solution is a variable bit allocation scheme whereby more bits are allocated to the sonically more important coefficients and fewer bits to the less important coefficients.

A key component of a compression algorithm is the development of a strategy for determining the comparative importance of the coefficients. It is here that perceptual coding enters the picture. The idea behind a perceptual coder is to exploit limitations of the human hearing system to determine the relative importance of different frequency components. Common experience indicates that sounds can 'mask' each other. For example, when two sounds occur simultaneously the louder noise will mask the softer, provided the two sounds are close to each other in frequency. There are many different types of masking effects. A perceptual coder attempts to exploit different types of masking to determine the portion of the spectrum that is inaudible and that does not need to be allocated any bits.

The situation is more complicated in that masking need not be complete: sound A may partially mask sound B, reducing the loudness of B but not totally hiding it. For this reason a perceptual coder calculates two curves—the sound pressure level for the signal at different frequencies, and the sound pressure level of the masking curve. For each frequency the masking curve identifies the maximum amplitude of the signal at that frequency that will be masked by other components of the signal. This level is often referred to as the level of just noticeable distortion (JND), or the sound pressure level at which distortion just begins to become noticeable (Jayant, Johnston & Safranek, 1993, p. 1385). In order to implement a variable bit allocation algorithm, one needs to possess both the masking curve and a curve representing the strength of the signal itself. The difference between these two curves can then be used to determine how many bits to allocate to each coefficient. If the strength of the signal is greater than the strength of the masker at a given frequency, then the distance between the two curves can be used to determine the number of bits that must be allocated to quantize the coefficient. A common approach is to allocate one bit for every six decibels of separation between the signal and the masker.

B. Threshold in quiet

In addition to the masking effects exerted by some parts of the signal on others, the threshold in quiet is also used to measure the masked portions of the signal. The threshold in quiet is a masking curve that measures the sound pressure level below which a sound is inaudible even in a completely quiet environment. This threshold varies with frequency. At 20 Hz. the threshold in quiet is roughly 60 dB. As the frequency increases the threshold declines until it reaches its lowest level of -5 dB at a frequency of roughly

3500 Hz., the area of the human ear's greatest sensitivity. At higher frequencies the threshold rises again, reaching levels in excess of 60 dB around 17 kHz (Zwicker & Fastl, 1990, pp. 18-19). Any portions of the sound spectrum that fall below the threshold in quiet are inaudible and can be discarded without any loss of sound quality.

C. Critical bands

The concept of a critical band is of enormous importance in the development of a perceptual coder. A critical band is a range of frequencies with the property that the masking effects of the frequencies on a sound falling within the band are very different from the masking effects of the same frequencies on a sound falling outside the band. A considerable amount of research has been devoted to identifying the number and extent of critical bands. One type of listening test employs two maskers having the same sound pressure level, and a masked noise lying in the center of the suspected critical band. One masker has a higher frequency than the noise and the other has a lower frequency. Initially the frequency separation between the three sounds is small, and all three lie within the same critical band. As the test proceeds, the frequencies of the two maskers are gradually shifted so that they become farther away from each other and from the noise in between them. Up to a certain frequency separation, the shift in the frequencies of the maskers has no effect on the level of noise that they can mask. Once a certain frequency separation is reached, however, the masking effect of the two maskers declines precipitously (Zwicker & Fastl, 1990, p. 137). As a result of repeated tests conducted by different researchers, a consensus has been achieved as to the number and bandwidth of the critical bands. There

Table 2 Critical Bands

Band Number	Lower Frequency (Hz)	Upper Frequency (Hz)
1	0	100
2	100	200
3	200	300
4	300	400
5	400	510
6	510	630
7	630	770
8	770	920
9	920	1080
10	1080	1270
11	1270	1480
12	1480	1720
13	1720	2000
14	2000	2320
15	2320	2700
16	2700	3150
17	3150	3700
18	3700	4400
19	4400	5300
20	5300	6400
21	6400	7700
22	7700	9500
23	9500	12000
24	12000	15500
25	15500	

are 25 critical bands; the frequency ranges covered by these bands are given in Table 2 (from Scharf, 1970).

It has been discovered that critical bands correspond to physical properties of the inner ear. Each critical band corresponds to a distance along the basilar membrane of 1.3 mm (Zwicker & Fastl, 1990, p. 144). Thus the masking properties of critical bands have a physical basis.

Examination of the differences between the bandwidths of the critical bands reveals that the relationship between the bandwidths cannot be understood in terms of either a

linear or a logarithmic scale. Consequently a special scale has been developed for describing critical bands—the bark scale. For instance, a frequency coefficient located one fourth of the way through the eighth critical band would have a bark value (with numbering starting at zero) of 7.25.

VI. Criteria for Evaluating a Perceptual Coder

Four main criteria are used in the evaluation of the effectiveness of a perceptual coding algorithm: compression ratio, sound quality, complexity and flexibility.

There is little that needs to be said about measuring the compression ratio that a particular coder achieves, since it can be measured easily. It is clear that there is a tradeoff between the compression ratio and the resulting sound quality of the decompressed file. Sound quality must be measured at a specified compression ratio. It is entirely possible that coder A will produce better sound quality than coder B at one compression rate, but that B will be preferable to A at a different rate.

The most difficult of the four criteria to implement is the assessment of sound quality. Both subjective and objective measures of sound quality have been suggested. Subjective measures of sound quality rely on listening tests to evaluate the quality of the decompressed file. One common scale used in these tests is the mean opinion score or MOS scale. This scale measures sound quality on five different levels—bad, poor, fair, good and excellent. Each adjective is associated with a number from one to five. Listening tests are repeated using many different subjects under varying conditions, and the results are averaged. Some researchers believe a negative scale is preferable—very annoying, annoying, slightly annoying, perceptible but not annoying, and imperceptible (Jayant,

Johnston & Safranek, 1990, p. 1388). Subjective measures of sound quality will not be considered further in this essay.

Before the advent of perceptual coders the traditional objective measure of sound quality was the signal to noise ratio. The signal to noise ratio is defined as $10 \log_{10}(\sigma_x^2 / \sigma_n^2)$, where σ_x^2 is the variance of the signal and σ_n^2 is the variance of the noise (Jayant & Noll, 1984, p. 6). This measurement, however, cannot be applied when a perceptual coding algorithm is used. The reason is the perceptual coder makes no attempt to minimize the noise in the signal; instead its goal is to distribute the noise in such a fashion that it is masked by other parts of the signal. A more appropriate mathematical measure of sound quality for a perceptual coder is the square of the unmasked noise. Later chapters will rely heavily on this measure.

The complexity of the codec is also of some significance. It is important to note, however, that complexity is primarily a problem in the decoder. For many applications the encoder does not have to run in real time. Therefore an algorithm that is able to shift complexity from the decoder to the encoder is to be preferred.

Finally, the flexibility of the coding algorithm should be evaluated. A 'flexible' coding algorithm is one that allows advances in perceptual coding to be incorporated into the encoder without requiring revision of the decoder. Since most actual audio decoders are implemented in hardware, it is advantageous to be able to make improvements to the encoding algorithm without the necessity of changing the decoder.

CHAPTER 2

THE MODIFIED DISCRETE COSINE TRANSFORM

I. Introduction

AC-3 uses a version of the Fourier transform known as the modified discrete cosine transform. An apparent limitation of the cosine transform in comparison with the standard Fourier transform is the loss of signal information. A Fourier transform contains both a sine component and a cosine component. Since the cosine transform contains only half as much information, it is not possible to reconstruct the original signal without taking special precautions. For instance, if a 512-point cosine transform is performed, the result is 512 frequency-domain coefficients. However, since every cosine value in the range from π to 2π has an equivalent in the range from 0 to π , only 256 of these coefficients are distinct and can be used to reconstruct the original signal. And yet these 256 coefficients do not contain enough information to reconstruct the original 512 coefficients.

The solution is to use each time-domain sample in two consecutive cosine transforms. To illustrate, let A, B... F each represent a group of 256 time-domain samples. A cosine transform can be performed on each of six consecutive blocks of 512 samples in the following manner (allowing for the padding of the first block with zeroes):

Block	Samples Used
0	0A
1	AB
2	BC
3	CD
4	DE
5	EF

Since each group of samples is used in two consecutive transform operations, enough spectral information is generated to allow for the reconstruction of the original signal.

This approach brings with it several sources of difficulty. The first difficulty—referred to hereafter as the ‘overlap-and-add problem’—is that it is necessary to determine the appropriate method of combining the coefficients in two consecutive blocks when performing the inverse transform. The second and more serious difficulty—the aliasing problem—arises because the transform operation introduces aliasing terms into the frequency coefficients. If the original signal is to be perfectly reconstructed, it is essential to find a method of eliminating these aliasing terms.

The remainder of this chapter describes in some detail the mathematical foundations of the modified discrete cosine transform. The discussion will serve both to elucidate the workings of this transform, which is in common use in audio compression, and to explain how the transform can be formulated so as to solve the overlap-and-add and the aliasing problems. The first several sections are based on the work of Princen and Bradley (1986), whose derivation of the cosine transform clarified the steps necessary to remove aliasing terms from the restored frequency coefficients.

II. The Forward and Inverse Fourier Transforms

Princen and Bradley begin by deriving the forward and inverse Fourier transform equations. The goal is to derive equations for the forward and inverse transforms, and substitute the former into the latter in order to determine whether the combination of the forward and inverse operations cancel each other out and restore the original coefficients. This sequence of steps is described below.

The formula for the forward Fourier transform is:

$$X_k(m) = \cos\left(\frac{m\pi}{2}\right) \sum_{n=-\infty}^{\infty} x(n)h(P-1+nM-n) \cos\left(\frac{2k\pi}{K}(n+n_0)\right) + \\ \sin\left(\frac{m\pi}{2}\right) \sum_{n=-\infty}^{\infty} x(n)h(P-1+mM-n) \sin\left(\frac{2k\pi}{K}(n+n_0)\right)$$

In this formula K is the block size, or the number of time-domain samples in a transform block. In the AC-3 coder discussed later K will normally be set to 512. The variable M is equal to $K/2$, and n_0 is a time shift the purpose of which will be explained more fully later. Finally, h is a digital filter that is non-zero only in the range $0 \leq n \leq P-1$. To simplify this formula two substitutions are made: $n = mM + r$, and $M = K/2$:

$$X_k(m) = \cos\left(\frac{m\pi}{2}\right) \sum_{r=0}^{P-1} x(mM+r)h(P-1-r) \cos\left[\frac{2k\pi}{K}\left(\frac{mK}{2}+r+n_0\right)\right] + \\ \sin\left(\frac{m\pi}{2}\right) \sum_{r=0}^{P-1} x(mM+r)h(P-1-r) \sin\left[\frac{2k\pi}{K}\left(\frac{mK}{2}+r+n_0\right)\right]$$

To simplify this expression further:

- 1) Let $x_m(r) = x(mM+r)$;

2) 2) Observe that $(2\pi k/K) * (mK/2) = \pi mk$. The cosine of $(\pi mk + x)$ is equal to $\cos(x)$ when mk is even and $-\cos(x)$ when mk is odd. An analogous statement is true in regard to the sine function. Therefore:

$$X_k(m) = (-1)^{mk} \cos \frac{m\pi}{2} \sum_{r=0}^{P-1} x_m(r) h(P-1-r) \cos \left[\frac{2k\pi}{K} (r+n_0) \right] +$$

$$(-1)^{mk} \sin \frac{m\pi}{2} \sum_{r=0}^{P-1} x_m(r) h(P-1-r) \sin \left[\frac{2k\pi}{K} (r+n_0) \right]$$

Note that for m even only the cosine term is non-zero and for m odd only the sin term is non-zero. This equation is a convenient formulation of the forward transform.

The next step is to derive a similar formula for the inverse transform. Since it has not yet been established that the inverse transform operation restores the original coefficient, the output of the inverse transform will be designated as x' . The formula for the inverse transform is:

$$x'(n) = \frac{1}{K} \sum_{k=0}^{K-1} \cos \left[\frac{2k\pi}{K} (n+n_0) \right] \sum_{m=-\infty}^{\infty} X_k(m) \cos \frac{m\pi}{2} f(n-mM) +$$

$$\frac{1}{K} \sum_{k=0}^{K-1} \sin \left[\frac{2k\pi}{K} (n+n_0) \right] \sum_{m=-\infty}^{\infty} X_k(m) \sin \frac{m\pi}{2} f(n-mM)$$

Most of the variables in this equation have already been discussed. In this equation f represents a windowing function; it has the same range of valid values as h . Changing the order of summations and substituting $r = n - m_0M$:

$$x'_{m_0}(r) = \sum_{m=-\infty}^{\infty} f(r+m_0M-mM) \cos \left(\frac{m\pi}{2} \right) \frac{1}{K} \sum_{k=0}^{K-1} X_k(m) \left[\frac{2k\pi}{K} (r+m_0M+n_0) \right] +$$

$$\sum_{m=-\infty}^{\infty} f(r+m_0M-mM) \sin \left(\frac{m\pi}{2} \right) \frac{1}{K} \sum_{k=0}^{K-1} X_k(m) \sin \left[\frac{2k\pi}{K} (r+m_0M+n_0) \right]$$

Let $y'_m(r)$ be the above expression with $\sum_{m=-\infty}^{\infty} f(r + m_0M - mM)$ factored out. Carrying out simplifications parallel to those used for the forward transform yields:

$$y'_m(r) = \cos\left(\frac{m\pi}{2}\right) \frac{1}{K} \sum_{k=0}^{K-1} (-1)^{mk} X_k(m) \cos\left(\frac{2k\pi}{K}(r + n_0)\right) + \\ \sin\left(\frac{m\pi}{2}\right) \frac{1}{K} \sum_{k=0}^{K-1} (-1)^{mk} X_k(m) \sin\left(\frac{2k\pi}{K}(r + n_0)\right)$$

$$\text{Thus } x'_{m_0}(r) = \sum_{m=-\infty}^{\infty} f((m_0 - m)M + r) y'_m((m_0 - m)M + r), \text{ for } r = 0, 1, \dots, M-1 \quad (1)$$

Recall that the length of the two windows f and h is P . Let $K/2 \leq P \leq K$. Given that $M = K/2$, it follows that (1) is zero except when $m = m_0$ and $m = m_{0-1}$.

$$\text{Thus } x'_{m_0}(r) = f(r + M) y'_{m_{0-1}}(r + M) + f(r) y'_{m_0}(r) \quad (2)$$

The point of this derivation is that when one performs a forward transform followed by an inverse transform, the original coefficient can be recovered from only two of the resulting terms, assuming appropriate windowing functions are employed.

III. The Forward and Inverse Discrete Cosine Transforms

In the previous section it was demonstrated that the inverse Fourier transform is able to recover the original frequency coefficients through a fairly straightforward method. In this section a parallel derivation is presented for the discrete cosine transform.³ The formula for the forward cosine transform is:

$$X_k = \sum_{n=0}^{K-1} x(n) \cos\left(\frac{2k\pi}{K}(n + n_0)\right) \text{ for } k = 0, 1, 2, \dots, K-1 \quad (3)$$

³ The material in this section is based on Princen & Bradley (1986, pp. 1157-1158).

The formula for the inverse cosine transform is:

$$x'(m) = \frac{1}{K} \sum_{k=0}^{K-1} X_k \cos\left(\frac{2k\pi}{K}(m+n_0)\right) \quad (4)$$

The goal is to determine whether the combined effect of the two transforms is to restore the original coefficients. Substituting (3) into (4) yields:

$$\begin{aligned} x'(m) &= \frac{1}{K} \sum_{k=0}^{K-1} \sum_{n=0}^{K-1} x(n) \cos\frac{2k\pi}{K}(n+n_0) \cos\left(\frac{2k\pi}{K}(m+n_0)\right) \\ &= \frac{1}{K} \sum_{k=0}^{K-1} x(n) \sum_{n=0}^{K-1} \cos\left(\frac{2k\pi}{K}(n+n_0)\right) \cos\left(\frac{2k\pi}{K}(m+n_0)\right) \end{aligned} \quad (5)$$

Some effort must now be expended to simplify this expression. Let S = the inner summation.

Using the identity $\cos(a)\cos(b) = \frac{1}{2}[\cos(a+b) + \cos(a-b)]$

$$\begin{aligned} S &= \frac{1}{2} \left[\cos\frac{2k\pi(n+n_0)+2k\pi(m+n_0)}{K} + \cos\frac{2k\pi(n+n_0)-2k\pi(m+n_0)}{K} \right] \\ &= \frac{1}{2} \left[\cos\frac{2k\pi(m+n+2n_0)}{K} + \cos\frac{2k\pi(n-m)}{K} \right] \end{aligned}$$

The next step is to find a way to simplify a summation where the term being summed is an expression involving a cosine. To this end let w represent some expression

containing integers. We wish to determine the value of $\sum_{k=0}^{K-1} \cos\frac{2wk\pi}{K}$

$$\sum_{k=0}^{K-1} \cos\frac{2wk\pi}{K} =$$

$$\frac{1}{2} \sum_{k=0}^{K-1} e^{\frac{-2\pi wjk}{K}} + \frac{1}{2} \sum_{k=0}^{K-1} e^{\frac{2\pi wjk}{K}}$$

$$\begin{aligned}
&= \frac{1}{2} \sum_{k=0}^{K-1} \left(e^{-\frac{2\pi w j}{K}} \right)^k + \frac{1}{2} \sum_{k=0}^{K-1} \left(e^{\frac{2\pi w j}{K}} \right)^k \\
&= \frac{1}{2} \frac{1 - e^{-2\pi w j K}}{1 - e^{-\frac{2\pi w j}{K}}} \\
&= \frac{1 - \cos(2wjk\pi) + j \sin(2wjk\pi)}{1 - e^{-\frac{2jw\pi}{K}}} \quad (6)
\end{aligned}$$

Since for any integer C, $\cos(2\pi C) = 1$ and $\sin(2\pi C) = 0$, the numerator in the above expression is always zero. Consequently $\sum_{k=0}^{K-1} \frac{2wjk\pi}{K}$ is zero for all values of (6) for which the denominator is defined. The constraints on the value of w in (6) are that $w \neq 0$ and $w \neq K$. Therefore $\sum_{k=0}^{K-1} \frac{2wjk\pi}{K} = 0$ when $w \neq 0$ and $w \neq K$. Returning to (5) and substituting the new value of S:

$$\begin{aligned}
x'(m) &= \frac{1}{K} \sum_{n=0}^{K-1} x(n) \left[\frac{1}{2} \sum_{k=0}^{K-1} \cos\left(\frac{2k\pi}{K}(m+n+2n_0)\right) + \frac{1}{2} \sum_{k=0}^{K-1} \cos\left(\frac{2k\pi}{K}(n-m)\right) \right] \\
&= \frac{1}{K} \sum_{n=0}^{K-1} x(n) \frac{1}{2} \sum_{k=0}^{K-1} \cos\left(\frac{2k\pi}{K}(m+n+2n_0)\right) + \frac{1}{K} \sum_{n=0}^{K-1} x(n) \frac{1}{2} \sum_{k=0}^{K-1} \cos\left(\frac{2k\pi}{K}(n-m)\right)
\end{aligned}$$

For the first cosine expression, $m + n + 2n_0$ can never equal zero. So this expression is non-zero only when $m + n + 2n_0 = K$. In this case the $\cos(m+n+2n_0) = 1$ and the summation equals K. For the second cosine expression, $n - m$ can never equal K. So this expression is non-zero only when $n - m$ equals zero (that is, when $n = m$). In this event the summation is again equal to K. Therefore:

$$x'(m) = \frac{1}{K} x(K - m - 2n_0) \frac{K}{2} + \frac{1}{K} x(m) \frac{K}{2}$$

$$= \frac{x(K-m-2n_0)}{2} + \frac{x(m)}{2} \quad (7)$$

Thus the recovered coefficient is equal to the original plus an aliasing term. The aliasing term—the first term above—is itself a shifted replica of the original, except that it is reversed in time. Once the value of $2n_0$ is set, (7) provides a means of solving the overlap-and-add problem.

IV. The Windowing Function

The derivation to this point has shown that the combination of a forward and inverse cosine transform recovers the original signal modified by an aliasing term. The final step is to show how the aliasing term can be removed. Princen and Bradley (1986, p. 1158) show that proper construction of the windowing functions can guarantee that the aliasing terms are cancelled and the original coefficient can be recovered exactly.

$$\text{First, let } g_m(r) = h(K-1-r)x_m(r) \quad (8)$$

The derivation of (7) is based on the cosine transform, for which the value of m_0 is even.

When m_0 is odd, (7) is replaced by

$$x'(n) = \frac{x(n)}{2} - \frac{x(K-n-2n_0)}{2} \quad (9)$$

Since the idea is to derive a window function that will cancel the alias, we want to replace $x(n)$ in (7) and (9) with (8), which includes a window. When (7) and (9) are combined and the windowing function is added, the result is:

$$y_{m_0}(r) = \frac{g_{m_0}(r)}{2} + (-1)^{m_0} \frac{g_{m_0}(K-r-2n_0)}{2} \quad (10)$$

For the cosine function m_0 is even. When m_0 is even, m_{0-1} is odd, yielding:

$$y_{m_0-1}(r+M) = \frac{g_{m_0-1}(r+M)}{2} - \frac{g_{m_0-1}(K-r-M-2n_0)}{2} \quad (11)$$

Substituting (1) into (2) yields:

$$x'_{m_0}(r) = f(r+M) \left[\frac{g_{m_0-1}(r+M)}{2} - \frac{g_{m_0-1}(K-r-M-2n_0)}{2} \right] +$$

$$f(r) \left[\frac{g_{m_0}(r)}{2} + \frac{g_{m_0}(K-r-2n_0)}{2} \right]$$

Next, observe that $h(m_0) = h(m_0-1)$. The reason is that m_0 is an entire block of coefficients, and the filters f and h only have a length of M , so they repeat for each block. With this observation in mind, and using (8) to substitute for g :

$$x'_{m_0}(r) = f(r+M) *$$

$$\left[\frac{h(K-1-r-M)x_{m_0-1}(r+M)}{2} - \frac{h(K-1-K+r+M+2n_0)x_{m_0-1}(K-r-M-2n_0)}{2} \right]$$

$$+ f(r) \left[\frac{h(K-1-r)x_{m_0}(r)}{2} + \frac{h(K-1-K+r+2n_0)x_{m_0}(K-r-2n_0)}{2} \right]$$

To simplify this expression note the following:

- (1) $x_{m_0-1}(r+M) = x_{m_0}(r)$ (because the block size of m_0 is equal to M).
- (2) $M = K/2$. Therefore $K - M = M$.
- (3) $x_{m_0-1}(K - r - M - 2n_0) = x_{m_0}(K - r - M - 2n_0 + M) = x_{m_0}(K - r - 2n_0)$.

These three observations yield:

$$x'_{m_0}(r) = f(r+M) \left[\frac{h(M-1-r)x_{m_0}(r)}{2} - \frac{h(M+r-1+2n_0)x(K-r-2n_0)}{2} \right]$$

$$+ f(r) \left[\frac{h(K-1-r)x_{m_0}(r)}{2} + \frac{h(r-1+2n_0)x_{m_0}(K-r-2n_0)}{2} \right]$$

Reorganizing yields:

$$x'_{m_0}(r) = x_{m_0}(r) \left[\frac{f(r+M)h(M-1-r) + f(r)h(K-1-r)}{2} \right] +$$

$$x_{m_0}(K-r-2n_0) \left[\frac{f(r)h(r-1+2n_0) - f(r+M)h(M+r-1+2n_0)}{2} \right]$$

The first term is the signal component, and the second term is an aliasing term. Perfect reconstruction of the signal can be achieved through the choice of appropriate windowing functions. More specifically, the first bracketed term must be equal to one, and the second bracketed term must be equal to zero. The following constraints are imposed on the windowing functions to ensure that the bracketed terms are equal to one and zero respectively:

$$1) f(r+M)h(M-1-r) + f(r)h(K-1-r) = 2, \text{ for } r=0 \dots M-1 \quad (12)$$

$$2) f(r)h(r-1+2n_0) - f(r+M)h(M+r-1+2n_0) = 0, \text{ for } r=0 \dots M-1 \quad (13)$$

While any windows that satisfy these constraints could be used, two additional constraints, although not strictly necessary, are useful:

3) $f = h$. That is, the transform window and the inverse transform window are the same.

4) $h(K-1-r) = h(r)$. In other words, the window is symmetric.

Using these additional constraints, (12) becomes:

$$f^2(r) + f^2(r+M) = 2 \quad (14)$$

And (13) becomes:

$$f(r) f(r - 1 + 2n_0) - f(r + M) f(M + r - 1 + 2n_0) = 0 \quad (15)$$

If $2n_0 = M + 1$, then (15) becomes $f(r) f(r+M) - f(r + M) f(r) = 0$, satisfying the window requirement.

In summary, it has been shown that the combined forward and inverse cosine transforms can exactly recover the original coefficients provided an appropriate window is used. The critical constraints on the choice of a window are embodied in (14) and (15).

V. A Generalized Windowing Function

The derivation of Princen and Bradley presented in the previous sections is limited by the fact that it presupposes a fixed block size—the value of the constant M . It is desirable in implementing the discrete cosine transform to be able to vary the block size. On the one hand large block sizes—1024 or even 2048 coefficients per block—are desirable because they provide enhanced frequency resolution. A large block brings with it the added benefit that, when the interval between frequencies is smaller, the loss in resolution by storing two coefficients as one value is reduced. On the other hand, the main disadvantage of large blocks is the generation of pre-echoes. When a sharp rise in the sound pressure level of the signal occurs near the end of the block, the digital filter in effect distributes this peak throughout the block, causing distortion. This problem disappears when the block size is short due to pre-masking: a loud sound masks noise that comes before it, provided the time interval is sufficiently short. Acoustic tests have shown that the pre-masking effect lasts only for a period of 20 ms.⁴ At a sampling frequency of 44100 samples per second, a block of 1024 samples would span a period of 23 ms. Therefore the block size must be shortened.

For these reasons audio coders typically include an option that allows the block size to be determined dynamically based on characteristics of the signal. It is consequently a matter of some importance to generalize the argument presented by Princen and Bradley to allow for variable block sizes. The derivation of this more general windowing function is presented below in some detail.⁵ In general the pattern of the argument follows Princen and Bradley.

The forward and inverse transform equations are:

$$X(k) = \sum_{n=0}^{N-1} x(n)h(n) \cos\left[\frac{2\pi}{N}\left(k + \frac{1}{2}\right)(n + n_0)\right]$$

$$y(m) = \frac{4}{N}h(m) \sum_{k=0}^{\frac{N}{2}-1} X(k) \cos\left[\frac{2\pi}{N}\left(k + \frac{1}{2}\right)(m + n_0)\right]$$

In these equations, k ranges from 0 to $N/2 - 1$; n ranges from 0 to $N - 1$; and n_0 is equal to $N/4 + 1/2$.

In order to determine the combined effect of the forward and inverse transforms, the values of $X(k)$ —that forward transform equation—is substituted into the inverse transform equation:

$$y(m) =$$

$$\frac{4}{N}h(m) \sum_{k=0}^{\frac{N}{2}-1} \sum_{n=0}^{N-1} x(n)h(n) \cos\left[\frac{2\pi}{N}\left(k + \frac{1}{2}\right)(n + n_0)\right] \cos\left[\frac{2\pi}{N}\left(k + \frac{1}{2}\right)(m + n_0)\right]$$

Reversing the summations:

⁴ See Section 4.2.

⁵ The derivation in this section is based on Iwadare, Sugiyama, Hazu, Hirano, & Nishitani (1992).

$$y(m) = \frac{4}{N} h(m) \sum_{n=0}^{N-1} x(n) h(n) \sum_{k=0}^{\frac{N}{2}-1} \cos \left[\frac{2\pi}{N} \left(k + \frac{1}{2} \right) (n + n_0) \right] \cos \left[\frac{2\pi}{N} \left(k + \frac{1}{2} \right) (m + n_0) \right] \quad (1)$$

Let S be the inner summation in (1). If n_0 is replaced by $N/4 + 1/2$, and we use the trigonometric identity $\cos a \cos b = 1/2 (\cos(a+b) + \cos(a-b))$, the value of S becomes:

$$S = \frac{1}{2} \sum_{k=0}^{\frac{N}{2}-1} \cos \left[\frac{2\pi}{N} \left(k + \frac{1}{2} \right) \left(n + m + 1 + \frac{N}{2} \right) \right] + \frac{1}{2} \sum_{k=0}^{\frac{N}{2}-1} \cos \left[\frac{2\pi}{N} \left(k + \frac{1}{2} \right) (n - m) \right] \quad (2)$$

In order to simplify S it is necessary to consider its parts separately. Let $A_1(n)$ be the first summation in S, and let $A_2(n)$ be the second summation in S.

Using the identity $\cos(a + b) = 1/2 (e^{j(a+b)} + e^{-j(a+b)})$:

$$A_2(n) = \frac{1}{2} \sum_{k=0}^{\frac{N}{2}-1} e^{j \frac{2\pi}{N} \left(k + \frac{1}{2} \right) (n-m)} + \frac{1}{2} \sum_{k=0}^{\frac{N}{2}-1} e^{-j \frac{2\pi}{N} \left(k + \frac{1}{2} \right) (n-m)} \quad (3)$$

Breaking this expression down further, let $B_1(n)$ be the first summation in $A_2(n)$, and let $B_2(n)$ be the second summation in $A_2(n)$.

$$\begin{aligned} B_1(n) &= e^{j \frac{\pi}{N} (n-m)} \sum_{k=0}^{\frac{N}{2}-1} e^{j \frac{2\pi k (n-m)}{N}} \\ &= e^{j \frac{\pi}{N} (n-m)} \sum_{k=0}^{\frac{N}{2}-1} \left(e^{j \frac{2\pi (n-m)}{N}} \right)^k \end{aligned}$$

Using the formula for the nth partial sum of a geometric sequence $\sum_{i=1}^N r^{i-1} = \frac{1-r^N}{1-r}$

$$B_1(n) = e^{j \frac{\pi}{N} (n-m)} \frac{1 - e^{j \frac{2\pi (n-m) N}{2N}}}{1 - e^{j \frac{2\pi (n-m)}{N}}}$$

$$= e^{j\frac{\pi}{N}(n-m)} \frac{1 - e^{j\pi(n-m)}}{1 - e^{j\frac{2\pi(n-m)}{N}}}$$

In parallel with the above derivation of $B_1(n)$ and omitting the details, it may be concluded that

$$B_2(n) = e^{-j\frac{\pi}{N}(n-m)} \frac{1 - e^{-j\pi(n-m)}}{1 - e^{-j\frac{2\pi(n-m)}{N}}}$$

Multiplying by $\frac{e^{j\frac{2\pi(n-m)}{N}}}{e^{j\frac{2\pi(n-m)}{N}}}$ and then by $-1/-1$ yields:

$$B_2(n) = -e^{j\frac{\pi}{N}(n-m)} \frac{1 - e^{-j\pi(n-m)}}{1 - e^{j\frac{2\pi(n-m)}{N}}}$$

Recall that $A_2(n) = \frac{1}{2} [B_1(n) + B_2(n)]$. Therefore

$$\begin{aligned} A_2(n) &= \frac{1}{2} e^{j\frac{\pi(n-m)}{N}} \left[\frac{1 - e^{j\pi(n-m)} - 1 + e^{-j\pi(n-m)}}{1 - e^{j\frac{2\pi(n-m)}{N}}} \right] \\ &= \frac{1}{2} e^{j\frac{\pi(n-m)}{N}} \left[\frac{-\{\cos(\pi(n-m)) + j \sin(\pi(n-m))\} + \cos(\pi(n-m)) - j \sin(\pi(n-m))}{1 - e^{j\frac{2\pi(n-m)}{N}}} \right] \end{aligned}$$

Since the numerator is zero, this equation is always zero. However, the expression is undefined for $n = m$. In order to find the value of A_2 when $n=m$, the substitution of $n=m$ into (3) shows that $\frac{1}{2} A_2 = N/4$.

A similar argument could be given to show (from 2) that $A_1(n) = 0$ when $n + m + N/2 + 1$ does not equal 0. Therefore it is necessary to find the value of $A_1(n)$ when $n + m + N/2 + 1 = 0$. Consider that $n + m + N/2 + 1 = 0$

$$\Rightarrow n = -N/2 - 1 - m$$

$$\Rightarrow n = N + (-N/2 - 1 - m)$$

$$\Rightarrow n = N/2 - 1 - m.$$

If this last value is substituted for n in the original expression for $A_1(n)$ (see (2)):

$$\begin{aligned} \frac{1}{2} A_1(n) &= \sum_{k=0}^{\frac{N}{2}-1} \cos \left[\frac{2\pi}{N} \left(k + \frac{1}{2} \right) \left(\frac{N}{2} - 1 - m + m + \frac{N}{2} + 1 \right) \right] \\ &= \sum_{k=0}^{\frac{N}{2}-1} \cos(2k\pi + \pi) = -N/2 \end{aligned}$$

$$\text{So } \frac{1}{2} A_1(n) = -N/4$$

Finally, substituting the values for A_1 and A_2 into (2):

$$y(m) = \frac{4}{N} h(m) \sum_{n=0}^{N-1} x(m) h(m) \frac{N}{4} + \frac{4}{N} h(m) \sum_{n=0}^{N-1} x \left(\frac{N}{2} - m - 1 \right) h \left(\frac{N}{2} - m - 1 \right) \frac{-N}{4}$$

Dropping the summations (since there is no longer an n term) and simplifying:

$$y(m) = h^2(m)x(m) - h(m)x \left(\frac{N}{2} - m - 1 \right) h \left(\frac{N}{2} - m - 1 \right)$$

In the previous block:

$$\begin{aligned} y' \left(m + \frac{N}{2} \right) &= h'^2 \left(m + \frac{N}{2} \right) x' \left(m + \frac{N}{2} \right) - \\ &h' \left(m + \frac{N}{2} \right) x' \left(\frac{N}{2} - m - \frac{N}{2} - 1 \right) h' \left(\frac{N}{2} - m - \frac{N}{2} - 1 \right) \end{aligned}$$

Given that

$$1) \quad h(m) = h(m + N/2);$$

$$2) \quad x'(m + N/2) = x'(m);$$

$$3) \quad x'(N/2 - m - 1) = x(-m - 1);$$

$$y'\left(m + \frac{N}{2}\right) = h^2\left(m + \frac{N}{2}\right)x(m) - h\left(m + \frac{N}{2}\right)x\left(\frac{N}{2} - 1 - m\right)h(N - 1 - m)$$

It was demonstrated earlier⁶ that the original coefficient can be exactly recovered by summing the inverse transform for the current coefficient and the corresponding coefficient in the previous block. Thus:

$$\begin{aligned} \underline{x''}(m) &= y'\left(\frac{N}{2} + m\right) + y(m) = \\ & x(m)\left[h^2(m) + h^2\left(m + \frac{N}{2}\right)\right] - \\ & x\left(\frac{N}{2} - 1 - m\right)\left[h\left(m + \frac{N}{2}\right)h(N - 1 - m) + h(m)h\left(\frac{N}{2} - 1 - m\right)\right] \end{aligned}$$

The conditions on the window, then, are that the first factor—the factor multiplied by $x(m)$ —must be equal to one, and the factor multiplied by $x(N/2-1-m)$ must be equal to zero. These conditions are not difficult to satisfy.⁷

In summary, the aliasing problem is solved by developing a window that satisfies the above constraints. The overlap-and-add problem is solved by summing two restored coefficients: $y'(N/2 + m)$ and $y(m)$. For example, consider a 512-point transform. When the inverse transform is performed on two consecutive blocks—blocks 0 and 1—the

⁶ Equation (7) in Section 2.3.

⁷ Iwadare, Sugiyama, Hazu, Hirano, & Nishitani (1992, p. 141) give one method for deriving the window values.

original coefficients are recovered by adding $\text{block0}[256] + \text{block1}[0]$, $\text{block0}[257] + \text{block1}[1]$, and so on.

CHAPTER 3

THE AC-3 CODEC

I. Introduction

Dolby's Audio Coder 3 (AC-3) has as its goal the encoding of 5.1 audio channels at a bit rate of 384 kbps (Advanced Television Systems Committee, 1995). The ".1" channel is the information for a subwoofer, which only operates at very low frequencies and so spans only a small portion of the bandwidth of audible sound. AC-3 supports sampling frequencies of 32, 44.1 and 48 kHz. Of these three sampling frequencies, the 44.1 kHz rate is currently the most important in that it is the standard for audio compact disks. The 44.1 sampling rate is the only rate that will be considered in this essay. The much lower sampling frequencies used in the processing of human speech are not supported by AC-3. The only way to use AC-3 to compress a sound file sampled at a lower rate is to first convert the file to a higher sampling rate. Consequently such files will not be compressed very efficiently.

It is important to note that Dolby's specification for AC-3 defines only the decoder, not the encoder. The purpose of this approach is to allow for future improvements in the encoder without having to modify the decoders. Since decoders are normally implemented in hardware, this flexibility is potentially a significant benefit. Of course, any future encoder must produce a compressed file that conforms to the decoder

specification. As mentioned earlier, the extent of the flexibility afforded to the encoder is an important issue in the evaluation of the algorithm as a whole.

The purpose of this chapter is to explain Dolby's algorithm in some detail. Emphasis will be placed on features of the algorithm that will be the subject of evaluation in subsequent chapters.

II. Overview of AC-3

A compressed AC-3 file consists of a series of 'frames'. A frame always begins with a Synchronization Information Header. This header contains information on the sampling frequency used in the file, but its primary purpose is to provide error checking. At a sampling frequency of 44.1 kHz, the sound samples in one frame represent only about $1/28^{\text{th}}$ of a second. Consequently if an error is detected in the file the decoder can mute for one frame and recover in the next frame without unduly affecting the final sound quality. The next component of the frame is a Bitstream Information Header. This lengthy header contains most of the variables that will control the decoding of the frame. To minimize the impact of an error information cannot be saved from one frame to the next, so these variables must be retransmitted with each frame. The content of some of the variables in this header will be discussed as the need arises.

A third component of a frame is a series of six audio blocks. These blocks contain the actual compressed sound information. For each block the encoder takes the PCM input samples and applies a windowing function and discrete cosine transform. Each block processes 512 time-domain samples into 256 frequency coefficients. It is these coefficients that are actually transmitted in the compressed file.

Within an individual audio block, the processing of the frequency coefficients proceeds by dividing each coefficient into an exponent and a mantissa. Since the coefficients are all real numbers between -1.0 and 1.0, the exponents are all negative powers of two. The exponents and mantissas are transmitted separately in the compressed file. The exponents are further compressed using a lossless algorithm.

The separation of the exponents from the mantissas is a crucial aspect of Dolby's solution to one of the central problems facing any audio coder. One of the keys to the success of an audio compression algorithm lies in the ability to allocate different numbers of bits to different coefficients. If an algorithm is to achieve a compression rate of 12:1 it is only possible to allocate an average of 1.5 to 2 bits to each coefficient. It is clear that if equal numbers of bits were allocated to all coefficients the resulting accuracy of the quantized values would be inadequate; hence the need for the variable bit allocation. The problem, however, is that once a variable bit rate scheme is adopted it is necessary to provide some mechanism to enable the decoder to determine where the bits for one coefficient end and the bits for the next coefficient begin. The method adopted to solve this problem is one of the most important factors shaping an audio compression algorithm.

Dolby's solution to this problem is rooted in the separation of the exponents from the mantissas. The exponents are transmitted in a lossless fashion, and the variable bit rate algorithm is used only to allocate bits to the mantissas. A perceptual coding algorithm is used to determine how many bits to allocate to each mantissa. This algorithm takes as its input not the original frequency coefficients but rather the compressed exponents. The decoder can completely recover the exponents and use them to rerun the perceptual coding algorithm in order to determine the bit allocation used for the compressed

mantissas. This clever solution profoundly shapes Dolby's entire algorithm. One of the main focuses of subsequent chapters is to examine the various tradeoffs entailed by this approach.

Once the variable bit allocation is calculated, further compression is applied to the mantissas. If the perceptual coder allocates zero bits to a mantissa no bits are transmitted for the mantissa. If the coder allocates more than five bits to a mantissa, the actual value of the mantissa is stored in the file. If the coder allocates from one to five bits, an offset into a lookup table is transmitted in the file. This approach makes it possible to achieve further compression gains. For instance, three mantissas that are allowed three quantization levels each can be grouped together and stored in five bits. Similarly, three mantissas that are allowed five quantization levels are stored in seven bits.

With this general overview as background, two components of the compression algorithm merit a more detailed study: the handling of the exponents and the perceptual coding algorithm. The latter is discussed in Chapter Four; the former is the topic of the remainder of this chapter.

III. Representation of Exponents

The first step in calculating the values of the exponents is to compute 'raw exponents' by removing the leading zeroes from the coefficients. An exponent represents an integer power of two, and should be interpreted as 2^{exp} .

In order to facilitate the lossless compression of the exponents, a set of 'differential exponents' is created by imposing three restrictions on the raw exponents: (1) the largest legal exponent is 24; (2) the first exponent in a block may not be larger than 15; and (3)

consecutive exponents cannot differ by more than two. The imposition of these restrictions will sometimes require adding leading zeroes back into the mantissas. For instance, if the first raw exponent in the block is 17, the differential exponent is set to 15 and two leading zeroes are added to the corresponding mantissa. The third condition is by far the most significant. When consecutive exponents differ by more than two, the strategy is always to decrease the larger exponent rather than to increase the smaller exponent. In view of the fact that small exponents correspond to large frequency coefficients, decreasing the larger exponent will usually avoid underquantization at the expense of overquantization. Since overquantization is clearly preferable to underquantization, this choice appears to be a reasonable one.

The use of the differential exponents assists in the compression of the exponents. The first exponent is stored as an absolute value. Only four bits are required because the first exponent was limited to a maximum value of 15. For the remaining 255 differential exponents, only the difference between the current and the previous exponent is transmitted. The differential values can range from -2 to +2. In order to make transmission more efficient each of these five values is increased to two, with the result that each exponent is represented by a number between 0 and 4. The 255 differential values are then stored in groups of three. If e_1 , e_2 and e_3 are three successive differential values, the stored value is $e_1 * 25 + e_2 * 5 + e_3$. In this manner three exponents can be stored with no loss of information using only seven bits.

In addition to the packing of three exponents into seven bits, further compression is achieved through the utilization of four alternative exponent 'strategies'. The first

strategy, called D15, is essentially the approach described above. That is, a single differential value is stored for each of the 256 exponents.

The second strategy--D25--stores differential exponents in pairs. The first exponent is still stored as an absolute value. For the remaining 255 exponents, each pair of exponents is represented by a single number. Thus the 255 original differential values are reduced to 128. The decision as to which of the two exponents to retain is always made in favor of the smaller exponent. The resulting 128 exponents are then packed together in triples as described above. These 128 exponents are still subject to the constraint that successive exponents can differ by at most two.

The third strategy—D45—is similar to D25 except that quadruples of exponents are combined instead of pairs. Thus the 255 differential exponents are grouped into 64 quadruples, with the smallest of the four exponent in each quadruple being retained. These 64 groups are then subjected to the constraint that successive exponents can differ by at most two.

The fourth strategy is to reuse the exponents from the previous block. This strategy cannot be used in the first block in a frame, because information is not retained across frames in order to minimize the propagation of error.

The choice between these four strategies is made dynamically by the program based on the characteristics of the signal. The AC-3 specification does not give any guidelines as to the kinds of considerations that should inform the choice of a strategy.

The tradeoff as one moves from D15 to D25 to D45 to Reuse is apparently the use of fewer bits to encode the exponents at the expense of a loss of resolution.⁸ The number of bits required to transmit information about the exponents and the exponents themselves can be summarized as follows. For all strategies, two bits are required to transmit a code identifying the strategy itself. For all strategies except reuse, an additional six bits are used to transmit a code for the number of exponent groups transmitted in the compressed file. This information is needed because many of the coefficients in the upper frequency ranges are often allocated no bits at all. By transmitting the number of groups it is possible to avoid transmitting 'trailing zeroes'—zeroes for the coefficients at the end of the block—for both exponents and mantissas. In addition to this general overhead, and assuming there are no trailing zeroes, the D15 strategy uses 599 bits ($4 + 85 * 7$) to transmit the exponents themselves, the D25 strategy uses 305 bits ($4 + 43 * 7$), and the D45 strategy uses 158 bits ($4 + 22 * 7$).

The transmission of data at very low bits rates imposes severe restrictions on the choice of exponent strategy. For example, assuming a sampling frequency of 44,100, it is necessary to use 28.7 frames to transmit the information associated with one second of sound (there are 1536 samples per frame). This means that for a target bit rate of 64 kbps, there are on average only 380 bits available to encode all the information contained in one audio block. It is apparent that this bit rate can be achieved only if the D15 and D25 strategies are used rarely, and that most blocks must reuse the exponents from the previous block.

⁸ It will be argued in Chapter 5 that this seemingly obvious statement is in fact seriously misleading.

CHAPTER 4

PERCEPTUAL CODING

I. Introduction

The goal of this chapter is to describe perceptual coding techniques and to develop a model for using these techniques to evaluate the effectiveness of the AC-3 coder. The argument of this chapter unfolds in the following manner. Section 2 summarizes the main conclusions of research into the masking effects of different types of sounds. Section 3 describes the AC-3 perceptual coding algorithm in detail, and Section 4 assesses the extent to which AC-3 implements the various types of masking properties outlined in the second section. The conclusion is that AC-3 implements only a small portion of the known masking properties of sound. As a result, if one is to measure the ability of the AC-3 coder to mask noise it is necessary to evaluate it in light of a different masking algorithm. With this goal in mind, Section 5 explains one of the masking algorithms that is used in the MPEG codec. Section 6 argues that, in light of the relative success of the AC-3 and MPEG masking algorithms in implementing the masking effects described in Section 2, it is appropriate to use the MPEG masking algorithm to measure the accuracy of the AC-3 masking calculation. The actual measurements are the subject of Chapter Five.

II. Types of Masking Effects

It will be recalled that the masking curve or threshold reflects the level of 'just noticeable distortion'—the sound pressure level of the test signal that is necessary for the signal to be audible in the presence of a masker. The masking effect may be either complete or partial. The existence of partial masking is demonstrated by the fact that as the level of a masker is gradually increased there is a continuous decrease in the loudness of the test signal until eventually it is completely masked (Zwicker & Fastl, 1990, p. 56).

There are several important types of masking effects. One type of masking effect is referred to as temporal masking, where the masker and the masked signal occur at slightly different moments in time. Temporal masking effects include both pre-masking and post-masking. The latter takes place when the masking tone occurs slightly before the masked sound, and the former occurs when the masked sound occurs slightly before the masker. The existence of post-masking is intuitively plausible: a loud sound will mask a softer sound that occurs just after it. However, pre-masking may seem less obvious—one might wonder how a masker can mask a signal that occurs before it in time. In spite of the fact that the pre-masking effect has been demonstrated experimentally, this phenomenon is currently not well understood. The suspected explanation is that a certain amount of time elapses between the point when a sound wave impacts on the ear and when the person is subjectively aware of the sound. If during this time interval another, louder sound enters the ear, it can interfere with the processing of the first sound. As one might suspect, post-masking lasts longer than does pre-masking. Experiments have shown that post-masking lasts for roughly 200 ms, whereas pre-masking lasts for only about 20 ms (Zwicker & Fastl, 1990, p. 73).

AC-3 does not explicitly implement temporal masking. However, the encoder is left undefined, and there is a mechanism available for telling the decoder to skip a block of coefficients entirely. While the primary purpose of this device is to react to error conditions in a file, it could also be used to implement temporal masking if one so desired. Since the specification does not deal directly with temporal masking, it will not be discussed further here.

The second main type of masking is frequency masking. The idea is to measure the masking effects when the masking and masked signals occur simultaneously. One important element of frequency masking is that the masking effects of sounds within a critical band are very different from the effects across critical bands. A second element is that the masking effect varies depending on whether the masker is a tone or a noise. Technically, 'noise' is any random element in a signal, and a 'pure tone' is any part of a signal that can be represented as a single sine wave (Luce, 1993, p. 16). It has been demonstrated experimentally that the efficacy of a masker depends on whether it is a tone or noise. In practice the distinction between tones and noise is implemented less rigorously. Any portion of the signal that contains sharp peaks in amplitude has masking properties similar to pure tones and is treated as a tone. And any portion of the signal that is relatively flat has masking properties similar to noise and is regarded as noise-like or non-tonal. These different types of maskers will now be examined in more detail.

A. Masking effects of noise

Various estimates have been given of the masking effects of a noise masker within a critical band. Jayant, Johnston and Safranek (1993, p. 1409), summarizing recent

research on the subject, suggest that the masking threshold produced by a noise is equal to the power of the noise diminished by a constant. The value of the constant has been estimated by different researchers as falling somewhere between three and six decibels, with their own recommendation being a value of five decibels.

Zwicker has probably conducted more extensive listening tests than anyone else, and his results are widely quoted in the literature. He differentiates between several types of noise that could serve as a masker. At low frequencies “white noise”—defined as a noise whose spectral density is independent of frequency—masks a signal at a level of 17 dB above the level of the noise itself. Above frequencies of 500 Hz, the masking threshold rises slowly at a rate of about 10 dB for every 10000 Hz (Zwicker & Fastl, 1990, pp. 57-58). Of greater significance for perceptual coding is ‘narrow-band noise’—noise occurring within a single critical band. Zwicker’s research suggests that the masking threshold decreases as the frequency of the noise increases. For example, given a narrow-band noise with a sound pressure level of 60 dB: if the center frequency of the noise is 250 Hz the masking threshold lies 2 dB below the noise; if the center frequency is 1 kHz the threshold lies 3 dB below the noise; and if the center frequency is 4 kHz the threshold lies 5 dB below the noise. In general, at very low frequencies the masking threshold is about 2 dB, and it rises to 6 dB at frequencies of 10 kHz and higher (Zwicker & Fastl, 1990, pp. 59, 153-154). These conclusions are of considerable significance in that they show the potency of a noise masker declines with increasing frequency. A difference of 4 dB corresponds to roughly 2/3 of a bit; a perceptual coder that ignores the decline in the masking effectiveness of noise at higher frequencies will underquantize at those frequencies.

It is important to observe that the effectiveness of a masker within a critical band depends on the location of the signal to be masked within the critical band. In particular, the full masking effect of the noise in the band will be realized only if the masked signal lies at the center of the band. As the masked signal moves further away from the center of the band the power of the masker diminishes.

B. Masking effects of tones

For tonal maskers, a common formula for the masking threshold could be given as:

$L_{\text{THRESH}} = L_{\text{MASKER}} - 14.5 - B$ (Jayant, Johnston & Safranek, 1993, p.1409). That is, the masking threshold is equal to the sound pressure level of the tonal masker minus 14.5 decibels minus the bark value of the masker. Other researchers have given the same formula (Johnston, 1988, p. 319).

This formula has several important implications. First, a tonal masker is less effective than a non-tonal masker. Recall that for a non-tonal masker the masking threshold lies between 2 and 6 dB below the level of the masker. For a tonal masker the distance between the masker and the masking threshold is much larger, indicating that a tonal masker is less effective. Second, as is the case with non-tonal maskers, the effectiveness of the masker diminishes as the critical band number increases. However, as frequency increases a tonal masker loses effectiveness much more rapidly than a non-tonal masker: whereas the efficacy of a non-tonal masker declines by only 4 dB, the efficacy of a tonal masker declines by 24 dB (the maximum value of B). Third, the effectiveness of a tonal masker does not remain constant within a critical band. A masker in critical band 8

may have a bark value of anywhere from 8.0 to 8.9, so the value of B is not constant even within a critical band.

C. The spreading function

A spreading function measures the loss of masking power as one crosses critical band boundaries. It is well established that a lower frequency can mask a higher frequency much more effectively than a higher frequency can mask a lower frequency. At intermediate frequencies the masking effect of band n on band $n-k$ decreases by roughly 25 decibels per band. In contrast the masking effect of band n on band $n+k$ decreases by only about 10 decibels per band. A more precise formula for the spreading function that is frequently cited is:

$$10\log_{10} B(x) = 15.81 + 7.5(x + 0.474) - 17.5\sqrt{1 + (x + 0.474)^2}$$

In this formula 'x' is the number of the critical band and $B(x)$ is the spreading function (Schroeder, Atal & Hall, 1979, p. 1648). The expression was derived from data developed by Zwicker.

Zwicker puts the lower slope of the spreading function—the masking effect of a higher band on a lower band—at roughly 27 dB. This slope remains constant at different frequencies (Zwicker & Fastl, 1990, p.152). The upper slope of the spreading function—the masking effect of a lower band on a higher band—is in practice close to constant across different frequencies, although his studies suggest that the slopes are somewhat higher at frequencies below 200 Hz. Zwicker's most important finding pertains to the effect on the slopes of an increase in the sound pressure level of the masker. For both tonal and non-tonal maskers the upper slope becomes more gradual as the sound pressure

level increases. At a sound pressure level of 100 dB, for example, the upper slope of the spreading function is only 5 dB. Curiously, Zwicker's findings indicate that for non-tonal maskers the lower slope decreases with increasing SPL, but that for tonal maskers the lower slope actually increases with rising SPL (Zwicker & Fastl, 1990, p. 64). Zwicker offers no explanation for this anomaly. The conclusion he draws is that changes in sound level will have little actual effect on the lower slope of the spreading function, since the tonal and non-tonal effects will more or less cancel each other out. However, there is a significant decline in the upper slope as the sound level rises.

D. Summary of masking effects

It will be convenient in subsequent sections to have a summary of the above masking effects. These effects can, with some loss of specificity, be distilled into the following six points:

1. A non-tonal masker is more effective than a tonal masker. While the masking curve will fall 2-6 dB below the level of a non-tonal masker, it will fall 14-38 dB below the level of a tonal masker.
2. The effectiveness of a masker within a critical band diminishes in proportion to the distance of the masked signal from the center of the band.
3. The effectiveness of both tonal and non-tonal maskers declines as the bark value increases. The effectiveness of tonal maskers declines at a higher rate than that of non-tonal maskers.

4. Masking is more effective within a critical band than across bands: the lower slope of the spreading function is about 25 dB per band; the upper slope is about 10 dB per band.
5. The spreading function does not depend on the bark value.
6. The upper slope of the spreading function becomes more gradual as the SPL of the masker—either tonal or non-tonal—increases.

III. Summary of the AC-3 Masking Algorithm

In general, a masking algorithm proceeds by calculating and then comparing two different curves. One curve, the power spectral density function, represents the strength of the signal at each frequency. The second curve, the masking curve, contains the level of just noticeable distortion, or the maximum signal strength that is masked at that particular point on the curve. If the signal strength is greater than the strength of the mask at a given point, then the distance between the two curves indicates the strength of the unmasked signal and determines how many bits need to be allocated to that coefficient. If the mask is greater than or equal to the signal, then the signal is completely masked at that point and no bits are allocated to the coefficient. This section describes in some detail how the AC-3 coder implements these general concepts. The implementation can be understood as proceeding in three stages: (1) calculation of the power spectral density functions for individual coefficients and for each band; (2) calculation of the excitation function, which represents the principal repository of the perceptual coding apparatus; and (3) the calculation of the final masking curve and bit allocation.

A. Power spectral density function

The first step in the masking algorithm consists of calculating the power spectral density function for each exponent. The power spectral density function of an individual coefficient is normally defined as the square of the magnitude of the coefficient (Jayant & Noll, 1984, p. 39). Since the cosine transform outputs real numbers, the square of the magnitude is equivalent to the square of the coefficient itself. Dolby has, however, chosen to use a slightly different approach. The power spectral density function of an individual exponent is given by: $\text{psd}[i] = 3072 - (\text{exp} \ll 7)$. The exponents are shifted left seven times to increase the resolution of the scale. The number 3072 was selected because it is the largest possible value on this scale—24 is the largest exponent, and $24 \ll 7$ is 3072. The reason the shifted exponent is subtracted from 3072 is that small exponents represent large coefficients; the subtraction simply ensures that large values on the psd scale correspond to large frequency coefficients.

AC-3 distributes the 256 frequency coefficients over 50 bands. Each pair of bands corresponds, at least roughly, to one critical band. A band psd is calculated for each band by summing the psds of the individual exponents within the band.⁹ The masking algorithm relies on the psd values for the bands, not the values for the individual exponents.

B. The excitation function

The excitation function is responsible for the actual calculation of the masking curve. The AC-3 specification includes pseudocode for the function, which must be

⁹ In reality what is summed are not the exponents but the numbers represented by the exponent and base together. The details are omitted here.

executed by the decoder. The crucial portion of this code is presented below in a somewhat modified form:

```

for (bands 7 to 21)

    lowcomp = calcLowcomp(lowcomp, bandPSD[band], bandPSD[band+1], band);

    fastleak -= fdecay;

    fastleak = max(fastleak, bandPSD[band] - fgain);

    slowleak -= sdecay;

    slowleak = max(slowleak, bandPSD[band] - sgain);

    excite[band] = max(fastleak - lowcomp, slowleak);

```

The calculation of the mask for bands 0-6 uses a considerably simpler approach, and will not be considered here. The calculation for bands 22-49 uses exactly the same code given above except that the lowcomp calculation is omitted.

This function embodies the heart of the AC-3 masking calculation. Consequently it is important to understand exactly what this function does and does not accomplish.

Passing over the reference to lowcomp for the moment, the code can be interpreted in the following manner. The values of four variables—fgain, sgain, fdecay and sdecay—may be set by values passed from the encoder to the decoder. The values are passed by table offsets, so there are only a small number of possible values for each variable. Alternatively, no values may be passed and the decoder can simply use defaults. If the default values for these variables are used and the lowcomp calculation is omitted, the code becomes:

```

fastleak = bandpsd[band] - 640

slowleak = bandpsd[band] - 1240

```

```

for (band = begin; band < bndend; band++)
    fastleak -= 83
    fastleak = max(fastleak, bndpsd[band] - 640)
    slowleak -= 19
    slowleak = max(slowleak, bandpsd[band] - 1240)
    excite[band] = max(fastleak, slowleak)

```

Consider first the fastleak calculation (the slowleak calculation is parallel). The value of fastleak is set to the maximum of (i) the band psd for the current band diminished by 640, and (ii) the current value of fastleak. The current value of fastleak, in turn, is the band psd from the previous band decreased by 640 plus a decay factor of 83.¹⁰ In other words, the value of the mask is set to either the band psd for the current band, decreased by 640, or to the band psd of the previous band, decreased by 640 plus 83, depending on which value is larger. Therefore fgain (or sgain) represents the masking power of the current band, and fdecay (or sdecay) represents the masking effect of the previous band(s) on the current band. It is critical to assess the significance of the particular values assigned to fdecay and fgain. This assessment will, however, have to be deferred until the rest of the perceptual coding algorithm is described.

The excitation calculation for the first 22 bands is adjusted based on the calcLowcomp function. In some cases the lowcomp computation will lower the level of the masking curve and therefore increase the bit allocation for the affected bands. The first part of this function reads as follows:

¹⁰ Assuming that the band psd of bin - 1 is greater than or equal to the band psd of bin - 2.

```

int calcLowcomp(int lowcomp, int psd0, int psd1, int band)
    if (band < 7)
        if ((psd0 + 256) = psd1)
            lowcomp = 384;
        else if (psd0 > psd1)
            lowcomp = (lowcomp - 64 > 0) ? lowcomp - 64 : 0;
    return lowcomp;

```

The remainder of the function is the same, with the exception that smaller values are assigned to lowcomp as the band number increases. The variables psd0 and psd1 contain the values of the band psd for the current band and the next band, respectively. The reason for testing whether $\text{psd0} + 256 = \text{psd1}$ is that 256 is the maximum possible difference between the psd values for these bands: consecutive exponents differ by at most two, and the first 28 bands only contain one coefficient each. This function, then, chooses between three options in setting the value of lowcomp: (1) if the exponent for the next coefficient (band) is two larger than the exponent for the current coefficient, set lowcomp to 384; (2) if the current exponent is greater than the next exponent, decrease lowcomp by 64; (3) otherwise leave lowcomp unchanged. Thus the function increases the value of lowcomp when the signal gets stronger, decreases the value of lowcomp when the signal gets weaker, and leaves the value of lowcomp the same when the signal is flat.

The lowcomp calculation is obviously intended to increase the number of bits allocated to the lower frequency bands. In one sense this maneuver is unjustified—there is no reason to believe that maskers are less effective at low frequencies; actually the reverse is true. However, at another level the strategy makes sense. If one is operating at a bit rate

that is sufficiently low to make underquantization inevitable, then it is better to underquantize at higher frequencies than at lower frequencies because of the human ear's greater sensitivity to lower frequencies.

C. Bit allocation

The third stage in the AC-3 perceptual coding algorithm is the calculation of the final masking curve and the actual bit allocation. The calculation of the masking curve adds several adjustments to the excitation function that complicate the analysis of that function's effect. There are four components that comprise this final stage.

First, there is a 'dbknee' adjustment to the excitation level. The value of this variable may be transmitted by the encoder to the decoder. The default value for dbknee is 2304, which corresponds to 2^6 . The excitation level for each band is adjusted according to the following formula:

```
if (bandpsd[band] < dbknee)
    excite[band] += ((dbknee - bandpsd[band]) >> 2)
```

By raising the masking curve for exponents greater than or equal to seven, this adjustment has the effect of increasing the bit allocation for frequencies with high sound pressure levels vis a vis those with lower levels.

Second, the excitation level for each band is compared with the threshold in quiet for that band. If the excitation level falls below the threshold in quiet it is raised up to the threshold.

Third, the excitation level is adjusted based on the value of the 'snroffset' variable. The value of snroffset is set based on the values of two variables passed to the decoder

according to the following formula: $\text{snroffset} = ((\text{csnroffset} - 15) \ll 4 + \text{fsnroffset}) \ll 2$.

These variables represent a coarse- and fine-grained modification of the masking curve respectively. The value of snroffset is subtracted from the excitation level to yield the final masking curve. If csnroffset is less than 15 then snroffset will be negative. In this case when snroffset is subtracted from the excitation level it will increase the level of the masking curve, thereby reducing the bit allocation. These variables in effect enable the encoder to tell the decoder to raise or lower the level of the entire masking curve by set increments.

Finally, once the masking curve is calculated it is possible to determine the number of bits to allocate to each coefficient. An offset into a lookup table is computed by subtracting the masking curve for the current band from the psd for an individual coefficient in the band, and then dividing this difference by 32. The lookup table contains 64 entries; the maximum number of bits that can be allocated is 15. There are approximately four entries in the table for each possible number of bits to be allocated, although the distribution of bits in the table is not completely uniform.

IV. Assessment of the AC-3 Masking Algorithm

The purpose of this section is to assess the extent to which the AC-3 masking algorithm succeeds in implementing the various types of masking effects discussed in Section 2 of this chapter. With this goal in mind, each of the masking effects mentioned in the masking summary in Section 2.4 will be reviewed in order to determine whether the AC-3 masking algorithm implements the effect.

Several features of the AC-3 masking algorithm complicate this effort. First, the results of the excitation function are in part reversed by the later dbknee and

snoffset computations. Second, there is no straightforward correspondence between the eventual bit allocation and the decibel value of the exponents. For example, it has already been mentioned that masking algorithms often allocate one bit to a coefficient for every six-decibel spread between the SPL of the signal and the SPL of the masking curve. On this basis, one might surmise that AC-3 would allocate a bit when the difference between the exponent and the masking curve is equal to 256 on the 3072-point scale (256 corresponds to two exponents, and one exponent corresponds to roughly three decibels). An examination of the bit allocation table, however, reveals that the correspondence is not so simple. Attempts are made to compensate for these difficulties in the ensuing discussion.

A. Intra-band tonal and non-tonal masking

The first masking effect refers to the superior masking capability of a non-tonal masker in relation to a tonal masker. It is evident from an examination of the excitation function that the AC-3 masking algorithm makes no attempt to distinguish between tonal and non-tonal maskers. It must be determined whether this omission leads to underquantization of the signal.

It will be recalled that the masking effect within a critical band is 2-6 dB for a non-tonal masker and $14.5 + B$ for a tonal masker, where B is the bark value of the critical band.¹¹ It would be helpful to derive similar numbers for the AC-3 masking effect within a band in order to facilitate a comparison. As a first step in this direction, it should be observed that the portion of the excitation function that pertains to masking effects within

¹¹ See 4.2 for details.

a band is the *fgain* variable.¹² The *fdecay* and *sdecay* variables only determine the masking effects of one band on a different band, and so can be ignored in the present context. The default value for *fgain* is 640. This value cannot be directly used to determine the masking power within a band, because the masking value implied in *fgain* is later modified by the *dbknee* and *snroffset* calculations. It is also necessary to consider how many bits AC-3 eventually will allocate for a given distance between the signal level and the level of the masker.

The computations reported in Tables 3 and 4 were carried out to address these difficulties. For purposes of these calculations, all 256 exponents in an audio block were set to a single value—1, 5, 10 or 15. The excitation function was modified so that it merely set the excitation level to the band *psd* minus *fgain*. With these exceptions, the remainder of the masking and bit allocation calculations were unchanged. Tables 3 and 4 contain the resulting bit allocations for selected bands. The values of 10 and 15 for *csnroffset* were chosen because tests show these to be reasonable minimum and maximum values: if *csnroffset* is set to a value less than 10 sound quality deteriorates dramatically, and a value greater than 15 severely compromises the compression ratio.

¹² The *sgain* variable is also related to masking within a single band; it is unnecessary to discuss *sgain* here.

Table 3 Bit Allocation (csnroffst = 10)

Band	Exp = 1	Exp = 5	Exp = 10	Exp = 15
1	3	3	2	0
5	3	3	2	0
10	3	3	2	0
15	3	3	2	0
20	3	3	2	0
25	3	3	2	0
30	2	2	1	0
35	2	2	1	0
40	1	1	1	0
45	1	1	1	0
50	0	0	0	0

Table 4 Bit Allocation (csnroffst=15)

Band	Exp = 1	Exp = 5	Exp = 10	Exp = 15
1	7	7	6	0
5	7	7	6	1
10	7	7	6	2
15	7	7	6	2
20	7	7	6	3
25	7	7	6	3
30	6	6	5	3
35	6	6	5	3
40	6	6	4	1
45	5	5	4	1
50	3	3	0	0

These tables indicate that the number of bits allocated to a particular coefficient due to masking effects within a single band may range from zero to 7. If one bit is taken to correspond to a distance of 6 dB between the signal and the masker, then it is apparent that the csnroffst setting provides the encoder with more than enough flexibility to meet the masking requirements within a single band. Assuming the value of csnroffst is set

sufficiently high, enough bits will be allocated to adequately reflect the masking effect within a single band.

The one surprising pattern that emerges in the above tables, however, is that the number of bits allocated to coefficients with the same sound pressure level decreases as the bark value increases. This problem will be discussed more extensively in Chapter Five.

B. Location within a band

The second masking effect is that the potency of a masker within a critical band diminishes as the masked signal moves farther away from the center of the critical band. The AC-3 coder does not take this effect into account. While it is true that the coder calculates a psd value for each individual coefficient, the excitation function only utilizes the psd for the critical bands. Consequently the location of an individual coefficient within a band is not taken into consideration. This approach will result in overestimating the strength of the mask for coefficients near the borders of critical bands, and consequently will underquantize these coefficients.

C. Masking at higher bark levels

The third masking effect is that the effectiveness of both tonal and non-tonal maskers declines as the bark value increases, with the effectiveness of tonal maskers declining at a higher rate. The AC-3 coder does not attempt to adjust for this phenomenon. On the contrary, the lowcomp calculation actually exacerbates the effect by increasing the bit allocation at lower frequencies. The primary effect of this choice is that the coder will underquantize at higher frequencies.

D. Spreading function: upper and lower slopes

The fourth masking effect is that masking is more effective within a critical band than across bands: the lower and upper slopes of the spreading function are 25 and 10 dB per band respectively. The AC-3 coder utilizes the `fdecay` variable to adjust for the upper slope of the spreading function. In particular, the band `psd` values for the present and the previous bands are adjusted downward by `fgain`, and the value for the previous band is adjusted down again by `fdecay`. The excitation level is set to the larger of these two adjusted values.

The problem, then, is to determine the relationship between the 10 dB decline suggested by psychoacoustic theory and the `fdecay` factor. The default value of `fdecay` is 83. When the masking curve is used to calculate the bit allocation, the difference between the signal strength and the strength of the masker is divided by 32. Since $83/32$ is roughly 2.5, the `fdecay` adjustment translates into approximately 2.5 slots in the bit allocation table. Since the bit allocation on average increases by one bit every four slots, and since one bit corresponds to 6 decibels, the `fdecay` adjustment is comparable to about 4 dB. Given that two AC-3 bands correspond to a single critical band, `fdecay` produces a decline of approximately 8 dB for each critical band. If this interpretation is correct, the default value of `fdecay` overestimates the masking effect of band n on band $n+1$, but only by a small amount.

In contrast, the AC-3 coder does not make any adjustment for the lower slope of the spreading function. The coder does not contain a look-ahead apparatus that would allow higher frequencies to mask lower frequencies. This decision is not particularly troublesome, however, because the masking effects of higher frequencies on lower

frequencies are minimal. In any event, the primary consequence is a small amount of overquantization; there is no resultant underquantization.

E. Spreading function and bark value

The fifth masking effect is that the spreading function does not depend on the bark value. The AC-3 coder encounters no difficulty here, since it does not adjust the spreading function in accordance with the bark value.

F. Spreading function and amplitude

The sixth masking effect is that the upper slope of the spreading function decreases as the sound pressure level of the masker increases. The AC-3 coder does not attempt to adjust for this effect. The effect is underestimation of the masking curve, and hence overquantization rather than underquantization.

G. Conclusions

The conclusion of this analysis is that AC-3 does not attempt to implement a masking calculation based on the details of perceptual masking. Part of the explanation for this state of affairs is that some of the masking effects cannot be implemented given other parts of the compression algorithm, especially the differential exponent strategy. In these cases, omission of some of the details of perceptual masking may be regarded as one of the tradeoffs necessary to achieve compression elsewhere in the coder. Another part of the explanation is that Dolby does not intend that the encoder rely on the excitation function alone to calculate the masking curve. Given that the specification does not define the encoder, there is no reason why the encoder cannot use a more sophisticated masking calculation to set the values of some of the variables it transmits to the decoder. In fact,

this seems to be precisely what Dolby has in mind: the specification suggests that the encoder set the values of the variables by trying different combinations iteratively until a combination is found that produces an acceptable masking curve for a particular block. The encoder can only carry out this task only if it has supplemental information about what the masking curve should be.

There are accordingly two reasons to search for another masking algorithm: it is necessary to have some standard that can be used to set the values of the various variables used by the decoder, and it is desirable to have some way of measuring the cumulative effect of the AC-3 masking algorithm on underquantization and overquantization. The remainder of this chapter examines one of the psychoacoustic masking algorithms put forward by MPEG, and evaluates its success in implementing the masking effects described earlier in the chapter.

V. The MPEG Masking Algorithm

MPEG is of course the best known alternative to Dolby's audio compression scheme. The two coders differ in numerous respects. No attempt will be made here to give a general comparison of the two approaches. The only portion of the MPEG algorithm that will be discussed in any detail is the masking calculation. MPEG 1 actually contains two perceptual masking algorithms. Only the first of these—called “Psychoacoustic Model 1”—will be described here (International Organization for Standardization, 1993).

A. Calculation of the power spectral density function

The MPEG coder does not split each frequency coefficient into an exponent and a mantissa. Consequently the coder uses the original frequency coefficients to perform the

masking calculation. A power spectral density value is calculated for each coefficient. Since MPEG, like AC-3, uses a modified discrete cosine transform, the coefficients are real numbers. The psd for a coefficient is calculated simply by converting the coefficient to a decibel value and normalizing to a maximum value of 96 dB. A psd value is also calculated for each critical band. This calculation is performed by summing the absolute values of the coefficients in each band, converting to a decibel scale and normalizing to a maximum level of 96 dB.

B. Identification of tonal and non-tonal maskers

MPEG labels each coefficient as either tonal or non-tonal. The tonal coefficients are identified first; any coefficient that is not tonal is classified as non-tonal. The identification of tonal components proceeds in three steps. First, each of the 256 frequency coefficients¹³ is examined to determine whether it is a local maximum. A coefficient $X(k)$ is a local maximum if $X(k) > X(k-1)$ and $X(k) \geq X(k+1)$. Second, a local maximum $X(k)$ is considered a tonal component if $X(k) - X(k+j) \geq 7$ dB. The variable j ranges over a series of positive and negative values. The exact range of j varies from ± 2 to ± 12 , depending on the frequency of the coefficient being examined. Third, the list of tonal components is decimated. That is, if there are two tonal components within 0.5 bark of each other, only the coefficient with the larger psd is retained. Finally each remaining tonal component $X(k)$ is stored using the formula:

¹³ While MPEG uses a cosine transform, it does not use a block size of 512. However the block size is not relevant to the present discussion, so it will be assumed that the block size is 512 and the MPEG masking algorithm is being applied to 256 coefficients.

$$X_{tm}(k) = 10 * \log \left[10^{\frac{X(k-1)}{10}} + 10^{\frac{X(k)}{10}} + 10^{\frac{X(k+1)}{10}} \right]$$

That is, the value stored is the sum of $X(k)$ and its two neighbors.¹⁴ There are at most two such values for each critical band.

A non-tonal level is also calculated for each band. A power spectral density function is calculated for the non-tonal coefficients in each band by summing the absolute values of the non-tonal coefficients in that band and converting to a decibel scale.

C. Calculation of individual masking thresholds

A masking threshold is calculated for each frequency coefficient. It is necessary to measure the extent to which each coefficient i is masked by other parts of the signal. Let $LT_{TM}[j, i]$ be the contribution of tonal masker j to the masking of i , and let $LT_{NM}[j, i]$ be the contribution of non-tonal masker j to the masking of i . The equations for calculating the contribution of a single masker to the masking of i are:

$$LT_{TM}[j, i] = X_{TM}(j) + av_{tm}(j) + vf(j, i)$$

$$LT_{NM}[j, i] = X_{NM}(j) + av_{nm}(j) + vf(j, i)$$

In these equations $X_{TM}(j)$ and $X_{NM}(j)$ represent the power of the tonal and non-tonal maskers, in decibels. The symbol 'av' is commonly used in the literature to refer to the masking index. It reflects the amount that the masking curve must be pushed down below the level of the masker j . This level is different for tonal and non-tonal maskers. The symbol 'vf' is an adjustment to the masking level based on the distance in bark between the masker and the masked coefficient. The vf calculation is identical for tonal and non-tonal maskers. Let us examine each of these variables in more detail.

The equations for the calculation of the masking index are:

$$(i) \quad av_{tm} = 1.525 - 0.275 * z(j) - 4.5$$

$$(ii) \quad av_{nm} = 1.525 - 0.175 * z(j) - 0.5$$

In these equations $z(j)$ is the value of the masker j on the bark scale. Computed av values for several values of j are included in Table 5. It is important to observe that the values in this table reflect the distance between the masker— X_{TM} or X_{NM} —and the masking curve, and not the distance between the signal and the masking curve. This distinction is important because the psd values for the signal are calculated by normalizing to a maximum level of 96 dB, while the maskers are calculated by normalizing to a maximum level of 48 dB. Consequently the distance between the signal and the masking curve is much greater than a cursory examination of Table 5 might suggest.

Table 5 Masking Index Values

j	av^{NM}	av^{TM}
0	1.025	-2.975
5	0.15	-4.35
10	-0.725	-5.725
15	-1.6	-7.1
20	-2.475	-8.475
24	-3.175	-9.575

In addition to distinguishing between tonal and non-tonal maskers, the av calculation also includes a decay factor that reduces the level of the mask as the bark value increases.

In calculating the value of vf , the distance in bark between i and j is stored in a variable dz , which is equal to $z(i) - z(j)$. If j is a tonal masker then $z(j)$ is the bark value of the coefficient j . If j is a non-tonal masker then $z(j)$ is the bark value of the non-tonal

¹⁴ The reason for the powers of 10 is that the $X(k)$ values are already on a decibel scale.

coefficient closest to the center of the band to which j belongs. Four different equations may be used to calculate the value of vf :

$$vf = 17 * (dz + 1) - (0.4 * X(j) + 6) \quad \text{for } -3 \leq dz < -1$$

$$vf = (0.4 * X(j) + 6) * dz \quad \text{for } -1 \leq dz < 0$$

$$vf = -17 * dz \quad \text{for } 0 \leq dz < 8$$

$$vf = -(dz - 1) * (17 - 0.15 * X(j)) - 17 \quad \text{for } 1 \leq dz < 8$$

The restriction of the calculation to critical bands in the range of $i-8$ to $i+3$ reflects the fact that bands farther away make too negligible a contribution to the masking curve to be worth considering. To illustrate the values generated by the vf equations, Table 6 reports some sample results with the value of the masked critical band i set to 10, and the value of the frequency of the maskers set to 10 and 50 dB.

Table 6 Spreading Function Values

dz	vf, x=10	vf, x=50
3	-110.0	-74.0
4	-94.5	-64.5
5	-79.0	-55.0
6	-63.5	-45.6
7	-48.0	-36.0
8	-32.5	-26.5
9	-17.0	-17.0
10	0.0	0.0
11	-10.0	-26.0
12	-27.0	-43.0
13	-44.0	-60.0

The table illustrates the fact that the upper slope of the spreading function is more gradual than the lower slope, and that the strength of the mask from one band to another diminishes at a slower rate when the frequency of the masker is higher.

D. The global masking calculation

A global masking value is computed for each coefficient i . The idea is that every coefficient that is within a distance of -8 or +3 bark from i is regarded as contributing to the masking of i , and the calculations outlined in the previous sections are made. The contributions of each of these coefficients are summed to form the global mask at i . An adjustment to the mask is also made to account for the threshold in quiet. The distance between the psd of the coefficient and the mask is then used to determine the bit allocation.

VI. Assessment of the MPEG masking algorithm

It will be useful to follow a procedure parallel to the one adopted in Section 4 in commenting on the AC-3 masking algorithm, and assess the MPEG algorithm in light of the masking effects described in Section 2.

The first masking effect is that a non-tonal masker is more effective than a tonal masker: while the masking index of a non-tonal masker is 2-6 dB, the index for a tonal masker is 14-38 dB. The MPEG coder clearly differentiates tonal from non-tonal maskers, and performs a calculation that reflects the superior masking power of non-tonal maskers.

One might be troubled by the fact that the numbers cited in Table 5 do not appear to correspond precisely to the masking indexes mentioned above. In particular, the masking index computed by MPEG is considerably smaller than the 14-38 dB suggested in Section 2. Two observations about these differences are in order. First, it should be recalled that the figures cited in Section 2 were based on the masking properties of noise and pure tones. In practice one must settle for an approximation by identifying 'tonal

components' of a signal based on the presence of a local peak. Second, it is important to recall that Table 5 measures the distance between the masking curve and the masker, not between the masking curve and the signal. Maskers are normalized to a maximum level of 48 dB, while signal components are normalized to a maximum level of 96 dB. By adopting this approach MPEG in effect guarantees that stronger portions of the signal will be allocated more bits relative to weaker portions of the signal. Consequently, depending of the strength of the signal at a particular point, the masking curve may be pushed down more or less than the numbers in Section 2 would require.

The second masking effect is that the effectiveness of a masker within a critical band diminishes in proportion to the distance of the masked signal from the center of the band. The MPEG coder fully adjusts for this phenomenon through the use of the bark scale. For tonal maskers the distance in bark between the masker and the masked coefficient is used as a multiplier in the v_f calculation to push the masking curve down further as the distance increases. For non-tonal maskers the psd of the masker reflects the strength of all the signals in a band, not merely one in particular. The calculation of the distance between the masker and the masked coefficient is accomplished by using the center frequency in the band as the bark value of the masker. Consequently the distance will increase as the masked signal moves further away from the center of the band.

The third masking effect is that the effectiveness of both tonal and non-tonal maskers declines as the bark value increases, and the effectiveness of tonal maskers declines at a higher rate than that of non-tonal maskers. The MPEG calculation of the masking index (av) incorporates this principle by decreasing the mask by a multiple of the bark value of the masker. For tonal maskers the bark value is multiplied by -0.275, and for

non-tonal maskers it is multiplied by -0.175 . Thus the masking curve declines as the bark value increases, and it declines more rapidly for tonal masker than for non-tonal maskers.

The fourth, fifth and sixth masking effects deal with the spreading function. The information presented earlier shows that the MPEG coder adjusts for these effects. Table 6 indicates that the coder computes a lower slope that is steeper than the upper slope, and that the sharpness of the upper slope decreases as the sound pressure level of the masker increases. Table 6 does suggest that, contrary to the information presented earlier, the sharpness of the lower slope computed by the coder also becomes less steep as the SPL of the masker increases. Given the small contribution of higher frequencies to the masking of lower frequencies, this effect will be small.

If one compares the analysis in this section with the analysis in Section 4, it is evident that the MPEG masking algorithm comes much closer to implementing known facts about perceptual masking than does the AC-3 masking algorithm. This statement is not, at this stage of the argument, intended as a criticism of AC-3. Any coding algorithm that hopes to achieve compression ratios in excess of 10:1 will have to make some compromises along the way. However, in light of MPEG's fuller realization of perceptual coding concepts, it is reasonable to use the MPEG masking algorithm to test the accuracy of the AC-3 algorithm. In other words, one can employ the MPEG algorithm to calculate an optimal bit allocation for an audio block, and then examine how close the AC-3 coder comes to realizing this allocation. This execution of this task is one of the main goals of Chapter Five.

CHAPTER 5

EVALUATION OF THE AC-3 CODER

This chapter has two major objectives. The first part of the chapter examines several attempts at improving the AC-3 algorithm that pertain to the handling of the exponents and to the masking algorithm. A study of these modifications shows that they are either unnecessary or that they fail due to interdependencies among different aspects of the algorithm. The second part of the chapter recommends one improvement to AC-3 that is useful under some circumstances. It is argued that the masking algorithm underquantizes high frequencies. Evidence of this underquantization is presented, and a solution to the problem is outlined.

A significant part of the argument in this chapter is derived from data generated by a test program that implements the AC-3 codec and the MPEG perceptual coder. The sample files were all .wav files sampled at 44.1 kbps.

I. Unsuccessful Revision Strategies

A. The exponent strategies

One area where there might appear to be room for improving the AC-3 coder is in the handling of the exponents. The reliance on differential exponents and the use of

exponent strategies other than D15 both introduce significant disparities between the raw exponents and the compressed exponents. The existence of this disparity takes on greater importance due to the central role played by the exponents in the bit allocation procedure.

As a first step in examining the exponents, it is illuminating to inquire into the amount of quantization error introduced by the differential exponents together with the four exponent strategies. Table 7 contains sample data from a test file containing a five second excerpt from Dvorak's "Slavonic Dance in C Major", taken from an audio compact disk. The table gives summary information for the 144 frames in the file. The first two columns—presented for purposes of comparison—contain the mean for the raw exponents and the variance of the raw exponents from the mean. The columns for D15, D25, D45 and REUSE express the error resulting from using the particular exponent strategy instead of the raw exponents. More specifically, the difference between each differential exponent and the original raw exponent is squared, and these values are summed to provide the total error for the block. The error for individual blocks is then averaged for the entire file. It should be stressed that trial runs on other test files produced similar results.

Table 7 Exponent Mean Squared Error

	Mean	Variance	MSE D15	MSE D25	MSE D45	MSE RSE
Block 0	12	4136	279	790	1241	-
Block 1	12	4135	261	769	1230	990
Block 2	12	4172	265	779	1232	1035
Block 3	12	4159	269	777	1240	1060
Block 4	12	4182	269	771	1237	1103
Block 5	12	4136	257	764	1216	1146

To some extent these numbers are what one would expect in that the error steadily increases as one changes strategy from D15 to D25 to D45. The main surprise in the table is the relatively small error produced by the REUSE strategy in relation to D45. In particular, for the first several blocks the REUSE error is significantly less than the D45 error, and for block 1 it is not too much higher than the D25 error.

Unfortunately these numbers are actually of fairly limited significance. The problem with this type of calculation is that a disparity between the raw and the differential exponents does not necessarily translate into a concomitant level of error in the bit allocation routine. In order to determine the relationship between the precision of the exponents and the accuracy of the bit allocation, tests were run to measure the quantization error consequent upon the use of each of the four different exponent strategies. The results for the same test file are contained in Table 8. For each test run the encoder was forced to use the same exponent strategy for every block in the file. Since the first block in a frame cannot reuse the previous exponents, for the reuse test the D15

strategy was used for the first block. The data on bit allocation is the average number of bits allocated per frame. To determine the overquantization and underquantization, the difference between the MPEG bit allocation and the actual number of bits allocated by AC-3 was squared.

Table 8 Exponent Strategies and Bit Allocation

Strategy	Bit Allocated	Overquantization	UnderQuantization
D15	1813	1250	753
D25	2084	2015	758
D45	2317	2550	685
REUSE	1816	1725	1461

One conclusion that can be drawn from the table is that selection of D15, D25 or D45 has very little bearing on underquantization, and therefore on the quality of the sound.

Contrary to what one might initially expect, moving from D15 to D25 to D45 actually reduces by a small amount the degree of underquantization. The explanation for this improvement in the underquantization numbers lies in the fact that when the D25 and D45 strategies select the exponent out of a group of two (or four) to be retained, they always select the smallest exponent in the group. The choice of the smallest exponent, or the largest coefficient, in effect overquantizes the sample, but as a result avoids underquantization. Consequently the choice between the first three strategies has very little bearing on sound quality. Of course, the amount of overquantization does increase significantly with the D25 and D45 strategies. Presumably, then, the encoder should

decide which of these strategies to use based simply on the calculation of whether the cost of transmitting the exponents is greater or smaller than the cost of overquantization. The reason for the dramatic increase in underquantization with the REUSE strategy is that this strategy contains no comparable forced overquantization mechanism.

At this point it is clear that error in the exponents—the difference between the raw exponents and the differential exponents—does not translate into a corresponding error in the bit allocation, at least insofar as underquantization is concerned. The implication is that attempts to improve the resolution of the exponents are not productive. For example, the fact that successive differential exponents can only differ by two limits the correlation between the differential and raw exponents. One way to try to improve the correlation is to use an adaptive differential scheme. The key to an adaptive scheme is that, while there are always only five values that can be used to quantize the differential exponents, there is no reason why these five values must always range from -2 to +2. If the current raw exponent is 1, a -2 is not possible for the next differential exponent. Furthermore, if the current raw exponent is significantly larger than the median it is probable that the next differential exponent will be negative rather than positive. An adaptive differential scheme, then, uses the -2 to +2 range only when the current exponent is close to the median for the block. If the current exponent is larger, the range is set to -3 to +1; if the current exponent is at the maximum the range is set to -4 to 0. Similar adjustments are made when the current exponent is smaller than the median. Test runs of a program demonstrate that this approach does indeed improve the correlation between the differential exponents and the raw exponents—the actual improvement in accuracy of the exponents is between 10% and 15%. However, the problem is that this modification does not actually reduce the

underquantization, and as a result it does not represent a meaningful improvement.¹⁵ The general conclusion, then, is that attempts to improve the correlation between the differential and raw exponents are not likely to be fruitful as they will not result in meaningful improvements in sound quality.

B. The masking algorithm

In the previous chapter it was shown that the AC-3 coder does not exploit many of the known masking properties of sound. There are, however, good reasons why some of these properties—in particular the distinction between tonal and non-tonal maskers, and the masking of lower frequencies by higher frequencies—are not taken into account. This section briefly comments on the reasons for these two omissions.

The overview of research into the masking effects of sound in the previous chapter plainly indicated that one of the principal aspects of masking is that tonal maskers are considerably less effective than non-tonal maskers. One might wonder why AC-3 does not take advantage of this information. However, the use of the differential exponents by the decoder to calculate the masking curve renders any adjustment for tonal and non-tonal components of the signal problematic. MPEG identifies tonal maskers primarily by testing whether a given frequency coefficient is at least seven decibels higher in level than its neighbors. A difference of two exponents is only a difference of about six decibels. Reliance on the differential exponents tends to smooth out the signal; there is no way for the decoder to determine whether a jump in an exponent of two is an indication of a much deeper valley in the signal. Furthermore, when the D45 or the REUSE strategies are

¹⁵ An additional problem with this modification is that if these modified exponents are reused in the next block, the correlation between the differential and raw exponents in the new block deteriorates.

adopted the correlation between the differential and raw exponents declines precipitously. To the extent that the differential exponents are not an accurate reflection of the actual exponents for the block, any attempt to exploit small changes in the differential exponents are unlikely to succeed.

Similarly, the AC-3 coder takes no account of the masking of lower frequencies by higher frequencies. This omission may be explained by the fact that the masking algorithm is run by both the encoder and the decoder. For example, suppose that for a certain audio block the last coefficient to actually be allocated any bits is in band 40. If the masking effects of higher frequencies on lower frequencies are considered, and if higher frequencies up to three critical bands above band 40 are allowed to mask the coefficients in band 40, then both exponents and mantissas for bands 41-43 would have to be transmitted even though they are all zero. To transmit data for these empty bands would not be justified, especially when one considers that there is a 25 dB per band decrease in the potency of a masker of higher frequency. It is worth emphasizing that this problem arises only because AC-3 requires the decoder to run the masking algorithm. If some other device were employed to inform the decoder of how bits had been allocated among the frequency coefficients, the problem would not occur.

The main consequence of the decision not to adjust for the two masking effects mentioned above is the overquantization of the file. Decisions that result in overquantization are inherently less objectionable than those that result in underquantization, since the latter do not adversely impact upon the quality of the sound. Moreover, if overquantization is the effect of tactics that save significant amounts of space elsewhere in the coder, then one might argue that the cost in extra bits is justified.

II. Underquantization of High Frequencies

There is one aspect of the masking algorithm that is more troubling than the features discussed above because it results in underquantization. In what follows it will be argued that the AC-3 masking algorithm underquantizes higher frequencies. This claim can be verified both in reference to the AC-3 masking algorithm and by means of test runs of the coder. These two lines of argument will be considered in turn.

The reason the AC-3 masker underquantizes higher frequencies is related to the fact that the power of both tonal and non-tonal maskers diminishes as the bark level increases. According to Zwicker, for non-tonal maskers the distance between the masker and the masking curve is about 2 dB at low frequencies and climbs to 6 dB at high frequencies.¹⁶ Jayant estimates that for tonal maskers this distance rises from 14 dB at low frequencies to around 38 dB at high frequencies.¹⁷ The review of the AC-3 masking algorithm in the previous chapter indicated that the algorithm makes no attempt to adjust for the declining power of maskers at higher frequencies, making underquantization of those frequencies inevitable.

The high frequency problem is exacerbated by another feature of the masking algorithm. Tables 3-4 in the preceding chapter indicate that when it is assumed that all exponents within an audio block are the same, and the excitation function is replaced with a simple fgain adjustment, more bits are allocated to the lower frequencies than to the higher frequencies. The reason is that the masking curve is calculated by subtracting fgain from the band psd. Whereas the lower bands contain only one frequency component each, the higher bands contain progressively larger numbers of coefficients. Therefore assuming

everything else is equal the masking curve will be higher in the upper frequency ranges. In one sense there is nothing objectionable about this procedure: to the extent that there are more coefficients in a critical band the masking effect will be stronger. However, this feature of the algorithm highlights the need to adjust for the declining power of maskers at higher frequencies. Summing the values within a band without making such an adjustment causes a significant amount of underquantization.

The second line of argument supporting the claim that AC-3 underquantizes the higher frequencies is a comparison of the output of a test program running both the MPEG and the AC-3 masking algorithms. The output is summarized in Tables 9-10 below. The tests were run by dividing the 256 coefficients into blocks containing 64 coefficients apiece. The first row in each table gives the number of bits the MPEG masking algorithm allocated in total and to each of the four blocks. The remaining rows state the number of bits allocated by AC-3 to each of the blocks, and the amount of underquantization and overquantization that was precipitated. Figures are provided for two different levels of the *csnroffst* variable. While the numbers of bits allocated are the actual totals, the underquantization and overquantization values were computed by taking the square of the difference between the number of bits MPEG allocated and the number of bits AC-3 allocated. The data in Table 9 is from a five second excerpt from a well-known jazz album, and was taken from an audio compact disk. The second file contains sound effects involving several scraping metals; it was sampled at 44.1 kHz.

¹⁶ See 4.2.a.

¹⁷ See 4.2.b.

Table 9 AC-3 Bit Allocation, Jazz Selection

	Csnroffst	Total	Block 1	Block 2	Block 3	Block 4
MPG Bits	-----	399756	184795	135993	69480	9488
AC3 Bits	10	179746	141511	31398	6690	147
UnderQ	10	651340	172727	277441	180407	20765
OverQ	10	97696	97559	136	1	0
AC3 Bits	14	477565	275924	147919	51077	2645
UnderQ	14	92997	6642	33354	39652	13349
OverQ	14	406694	351417	49192	5973	112

Table 10 AC-3 Bit Allocation, Scraping Metals

	Csnroffst	Total	Block 1	Block 2	Block 3	Block 4
MPG Bits	-----	36965	22616	6785	5485	2079
AC3 Bits	10	24340	22119	1578	609	34
UnderQ	10	49812	16034	12705	15326	5747
OverQ	10	17913	17857	30	26	0
AC3 Bits	14	62387	47770	9402	4785	430
UnderQ	14	12534	1026	2398	4795	4315
OverQ	14	101392	92896	5897	2469	40

It is apparent from both tables that AC-3 has significantly underquantized the upper frequencies. In Table 9, for example, with csnroffst set to 14 the total number of bits allocated by AC-3 was about 80000 more than the total number of bits allocated by MPEG. In spite of this difference, AC-3 allocated only 28% as many bits to the fourth block as did MPEG. For the same portion of Table 9, the ratio of the power of the underquantization to the number of bits allocated was 5:1 for the fourth block, but only 1:41 for the first block. Similarly almost all the overquantization occurred in the lower

blocks, and almost none occurred in the fourth block. The numbers in Table 10 show an even more dramatic bias in favor of the lower frequencies. When `csnroffst` was set to 14, AC-3 allocates almost twice as many bits overall as did MPEG, and yet AC-3 allocated only 1/5 as many bits to the fourth block.

In fairness, it must be pointed out that many sound files contain virtually no information in the higher frequency ranges, and in this sense the above examples are not typical of sound files in general. On the other hand, the numbers in the tables actually underestimate the high frequency problem in that they provide averages over whole files. When the bit allocations for individual blocks are examined, AC-3 underquantizes the upper frequencies in some blocks by much larger margins. One may conclude that AC-3's comparative neglect of the high frequency ranges will produce a perfectly acceptable result for some files, but for others it will create a significant problem.

III. Modifications to the Default Bit Allocation

One might legitimately wonder whether AC-3 itself contains resources for resolving the high frequency loss problem. In fact, AC-3 supplies the encoder with two mechanisms for instructing the decoder to modify the default bit allocation. The question arises, then, as to whether either of these mechanisms could be used to improve the quantization of higher frequencies. In this section these two mechanisms are presented, and it is argued that neither provides a satisfactory remedy for the high frequency problem.

The first tool for modifying the default bit allocation is the use of `csnroffst` to push the entire masking curve down by a set amount. However, while `csnroffst` furnishes the encoder with a powerful device for increasing the bit allocation across the entire spectrum,

it does not enable the encoder to selectively increase the bit allocation to some portion of the spectrum. An examination of Tables 9-10 demonstrates the limitations of `csnroffst` as a tool for solving the high frequency loss problem. While it is true that the higher value of `csnroffst` reduces the amount of underquantization in block 4 in absolute terms, the amount of underquantization in the fourth block actually increases when viewed as a percentage of the total amount of underquantization. For instance in Table 9, when `csnroffst` is set to 10, 3.2% of the underquantization occurs in block 4; when `csnroffst` is set to 14, 14.4% of the underquantization is in the fourth block. And in Table 10, when `csnroffst` is set to 10, 11.5% of the underquantization occurs in block 4; when it is set to 14, the percent of underquantization in the fourth block increases to 34.4%. Therefore a disproportionate amount of the benefit that comes from increasing the overall bit allocation goes to the blocks containing lower frequencies. It seems fair to conclude, then, that while this approach may reduce the severity of the underquantization of higher frequencies it is not an efficient remedy for the problem.

The second mechanism that AC-3 provides for modifying the default bit allocation is called a 'delta bit allocation'. The encoder may inform the decoder to increase the number of bits allocated to certain specific frequency ranges by a specified amount. At first glance this approach would appear to be precisely what is needed in that it allows the encoder to increase the number of bits allocated to certain frequencies without increasing the allocation across the entire spectrum. Unfortunately, an examination of the details of the delta bit allocation show that it is not suitable for use in the upper frequency ranges.

The delta bit allocation enables the encoder to instruct the decoder to increase the bit allocation for up to eight 'segments'. Each segment consists of a group of consecutive

bands for which the default bit allocation is to be increased. To carry out the delta bit allocation the encoder needs to transmit the following pieces of information: the number of segments used (3 bits); the number of the first band in the segment (5 bits); the number of bands in the segment (4 bits); and the number of bits to add to each coefficient in the segment (3 bits). Thus the overhead associated with this approach is fairly high, amounting to $3 + 12n$ bits, where n is the number of segments.

The fundamental problem with using a delta bit allocation to solve the high frequency loss problem is that the smallest unit the delta allocation can access is one band. There is no possibility of increasing the bit allocation for some of the coefficients within a band without increasing the allocation for all the coefficients in the band. This drawback may be acceptable when dealing with bands at lower frequencies, but the upper frequency bands contain too many coefficients for this solution to be viable. For example, bands 45-49 cover the coefficients from 133 to 255, and each of these bands contains 24 coefficients. A sound file normally consists of a series of alternating peaks and valleys. The peaks do not generally span a large number of coefficients, especially the peaks that occur in the high frequency range, which are normally small. Therefore, when one considers both the overhead associated with the delta bit allocation and the inability to augment the bit allocation for particular coefficients within a band, one must conclude that this strategy cannot be efficiently used to remedy the loss of high frequency information.

IV. A Remedy for High Frequency Loss

The argument thus far has attempted to demonstrate that AC-3 underquantizes at high frequencies and that there is no adequate corrective available within the confines of

the AC-3 specification. This section will propose a modification of the specification in order to resolve the problem.

Based on the discussion in previous sections, it is possible to identify the primary concerns that a successful solution should address. First, a supplemental high frequency bit allocation is not needed for all sound files. The encoder should have at its disposal a high quality perceptual coding algorithm that can be used to evaluate the need for allocating additional bits at high frequencies; the allocation should be made only as needed. Second, since signal peaks tend to span only a small number of coefficients at a time, it is essential to be able to allocate additional bits to only a small number of coefficients, and the allocation should not incur a high penalty in the form of additional overhead. Third, the encoder should be able to control how many additional bits are allocated.

The proposed solution is to use the exponents to identify peaks in the high frequency area. A 'peak' is a series of consecutive exponents that contain at least one coefficient with a high sound pressure level. Tests suggest that a 'high SPL' can be defined as an exponent of nine or smaller. Once a high SPL coefficient is identified, the peak is extended in each direction around the coefficient until an exponent is encountered which is larger than eleven. All such peaks are identified, with precautions to ensure that peaks do not overlap. The list of peaks is sorted by their 'power'. The power of a peak is calculated based on the exponents in the peak greater than or equal to nine, with smaller exponents being given increasingly greater weight. The encoder transmits a number indicating how many peaks to quantize. The decoder then allocates additional bits to the specified number of peaks, beginning with the peak of greatest power. Only coefficients in the peak having exponents less than or equal to ten are allocated any additional bits.

Table 11 Modified Bit Allocation: Jazz Selection

	Csnroffst	Total	Block 1	Block 2	Block 3	Block 4
MPG Bits	-----	399756	184795	135993	69480	9488
AC3 Bits	10	181659	141511	31398	6739	2011
UnderQ	10	643418	172727	277441	180180	13070
OverQ	10	97831	97559	136	1	135
AC3 Bits	14	479479	275924	147919	51126	4509
UnderQ	14	88263	6642	33354	39617	8650
OverQ	14	407797	351417	49192	6017	1171

Tables 11-12 report the results of test runs using this modified version of AC-3 on the two files analyzed earlier. The tests were conducted with `csnroffst` set to 10 and to 14. For the jazz file the number of bits allocated to the fourth block increased by a factor of 13 when `csnroffst` was set to 10, and by 70% when `csnroffst` was set to 14. For both `csnroffst` settings the power of the underquantized coefficients decreased by roughly 35%. The increase in the power of the overquantized bits was far smaller, making the tradeoff a reasonable one. The results for the metals file are similar. Even with `csnroffst` set to 14 the number of bits allocated to the fourth block was three times greater than with the default algorithm. For both `csnroffst` settings the power of the underquantized coefficients was cut more than in half, with much smaller increases in overquantization.

Table 12 Modified Bit Allocation: Scraping Metals

	Csnroffst	Total	Block 1	Block 2	Block 3	Block 4
MPG Bits	-----	36965	22616	6785	5485	2079
AC3 Bits	10	25306	22119	1578	621	988
UnderQ	10	46356	16034	12705	15262	2355
OverQ	10	18331	17857	30	26	418
AC3 Bits	14	63353	47770	9402	4797	1384
UnderQ	14	9876	1026	2398	4767	1685
OverQ	14	102078	98986	5897	2471	724

The usefulness of this rather simple modification depends of course on the target bit rate. If in a given context the goal is to achieve the lowest possible bit rate, then the default bit allocation—with *csnroffst* set to a low value such as 10—provides a sensible option. Under these circumstances there would be little point in adding bits at the higher frequencies. However, the AC-3 specification takes its target bit rate to be 384 kbps for six channels. Taking into consideration that four of the channels are coupled stereo channels and that one channel has a very narrow bandwidth, this target translates into approximately 75-80 kbps per full-bandwidth channel. Given that there are 28.7 frames per second, the target translates into a bit rate of approximately 2800 bits per frame. By way of comparison, the bit rates per frame for the jazz file with the modified bit allocation are 1253 and 3307 (with *csnroffst* values of 10 and 14 respectively). It is therefore possible to use the modified bit allocation algorithm and still attain the target bit rate.

It seems, then, that the peak enhancement strategy is a viable solution to the high frequency loss problem. In view of the high overhead associated with the delta bit allocation, it would be worth inquiring whether peak enhancement might be preferable to the delta bit allocation even in the lower frequency ranges.

BIBLIOGRAPHY

- Advanced Television Systems Committee. (1995). Digital audio compression standard (AC-3).
- International Organization for Standardization. (1993). Information technology—coding of moving pictures and associated audio for digital storage media at up to about 1.6 Mbit/s. Part 3: Audio. (ISO 11172-3).
- International Telecommunication Union. (1988). Pulse Code Modulation (PCM) of voice frequencies (Recommendation G.711). Geneva, Switzerland.
- Iwadare, M., Sugiyama, A., Hazu, F., Hirano, A., & Nishitani, T. (1992). A 128 kb/s hi-fi audio codec based on adaptive transform coding with adaptive block size MDCT. IEEE Journal on Selected Areas in Communications, 10, 138-144.
- Jayant, N., & Chen, E. Y. (1995, March/April). Audio compression: technology and applications. AT&T Technical Journal, pp. 23-33.
- Jayant, N., Johnston, J., & Safranek, R.. (1993). Signal compression based on models of human perception. Proceedings of the IEEE, 81, 1385-1421.
- Jayant, N., & Noll, P.. (1984). Digital coding of waveforms: Principles and applications to speech and video. Englewood Cliffs, NJ: Prentice-Hall.
- Johnston, J. D. (1988). Transform coding of audio signals using perceptual noise criteria. IEEE Journal on Selected Areas in Communications, 6, 314-323.
- Luce, R. D.. (1993). Sound and hearing: A conceptual introduction. Hillsdale, NJ: Lawrence Erlbaum.
- Princen, J. P., & Bradley, A. B. (1986). Analysis/synthesis filter bank design based on time domain aliasing cancellation. IEEE Transactions on Acoustics, Speech, and Signal Processing, 34, 1153-1161.
- Proakis, J. G., & Manolakis, D. G. (1992). Digital signal processing: Principles, algorithms, and applications (2nd ed.). New York: Macmillan.

- Scharf, B. (1970). Critical bands. In J. V. Tobias (Ed.), Foundations of modern auditory theory (Vol. 1, pp. 159-202). New York: Academic.
- Schroeder, M. R., Atal, B. S., & Hall, J. L. (1979). Optimizing digital speech coders by exploiting masking properties of the human ear. Journal of the Acoustical Society of America, 66, 1647-1652.
- Williams, J. E., Trinklein, F. E., Metcalfe, H. C., & Lefler, R. W. (1979). Modern physics. New York: Holt, Rinehart.
- Zwicker, E., & Fastl, H. (1990). Psychoacoustics: facts and models. Berlin: Springer-Verlag.