Chapter 3

How is sound processed in an MP3 player?

T. Dutoit (°), N. Moreau (*)

(°) Faculté Polytechnique de Mons, Belgium
(*) Ecole Nationale Supérieure des Télécommunications, Paris

Mr. Audio was a bit loose in his PCM tuxedo. Tayloring the suit helped it lose bits and become a lighter MP3 dinner jacket.

In his 1929 painting "La trahison des images," Belgian painter René Magritte highlighted the power of illusions by painting a pipe and commenting it with "Ceci n'est pas une pipe" ("This is not a pipe") as Magritte himself explained: "Try to stuff the painting with tobacco... ."

Perceptual illusions have been used by engineers for designing many consumer products. As an example, one of the most extraordinary imperfections of our visual process, known today as the phi effect (and mistaken for a long time with the persistence of vision), has induced the choice for 24 images per second in the movie pictures industry. In this chapter, we will see how the limitations of the human *auditory* process have been taken into account for the design of *lossy but transparent* perceptual audio coders. In particular, we will focus on the principles underlying the MPEG-1 Layer-I audio coding norm.

3.1 Background – Sub-band and transform coding

Sub-band signal processing is very useful in audio (as well as in video) coding applications. The sub-band encoder (Fig. 3.1) decomposes an input

T. Dutoit, F. Marqués (eds.), *Applied Signal Processing*,
DOI 10.1007/978-0-387-74535-0_3,

signal $x(n)$ into M different frequency ranges called *sub-bands* through the *analysis filter bank*. Sub-band signals $x_i(n)$ are then downsampled to $y_i(m)=x_i(Mm)$ and quantized. The decoder combines the quantized sub-band signals $\tilde{y}_i(m)$ into an output signal $\tilde{x}(n)$ by upsampling them to $\tilde{x}_i(n)$ (with $\tilde{x}_i(mM)=\tilde{y}_i(m)$ and $\tilde{x}_i(mM+l)=0$ for $l=0,...,M-1$) and passing them through the *synthesis filter bank* (Crochiere and Rabiner 1983, Vaidyanathan 1993). The number of sub-bands and the decimation–interpolation ratio are not independent; in a *critically sampled* filter bank, for which the number of sub-band signal samples is equal to the number of input samples, the number of sub-bands and the decimation–interpolation ratio are identical.

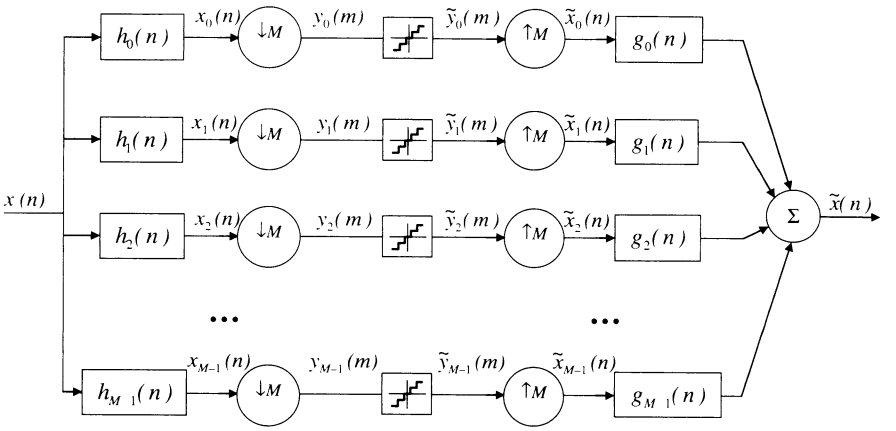

Fig. 3.1 A sub-band coder

The rationale for sub-band signal processing is the need for time- and frequency-dependent SNR control to allow transparent encoding of audio and video signals with very low bit rates. As a matter of fact, when applying uniform quantization to the input signal using a given number of bits per sample, the resulting quantization noise is uniformly distributed in the frequency range of the signal (see Chapter 2). Since the power spectral density (PSD) of the input signal changes with time and frequency, uniform quantization results in time- and frequency-dependent signal-to-noise ratio (*local* SNR). The number of bits per sample (hence the *overall SNR*) is therefore usually set to a very high value (the 16 bits–96 dB of the compact disk) in order for the worst-case local SNRs to be statistically acceptable to the human ear. In contrast, in sub-band signal processing, the

number of bits allocated to each sub-band signal can be adapted to the specific SNR required in the corresponding frequency band. This, as we shall see, allows for drastic bit rate reduction.

In this section, we will first examine the conditions required on the analysis–synthesis filters to satisfy the perfect reconstruction condition (Section 3.1.1). We will show that the filter bank approach can also be interpreted in terms of block transform-based processing (Section 3.1.2). The masking properties of the human ear will then be briefly reviewed, and their use in the MPEG1 – Layer I audio coding norm will be explained (Section 3.1.3). We will conclude by mentioning current coding norms and bit rates (Section 3.1.4).

3.1.1 Perfect reconstruction filters

An important feature of the analysis and synthesis filters in a filter bank is that they should allow *perfect reconstruction* (PR) (or, in practice, *nearly* perfect reconstruction) when no quantization is performed, i.e.,

$$\tilde{x}(n) = x(n) \tag{3.1}$$

In order to examine this problem in a simple case, let us focus on a two-channel filter bank (Fig. 3.2) composed of a low-pass (upper) and a high-pass (lower) branch.

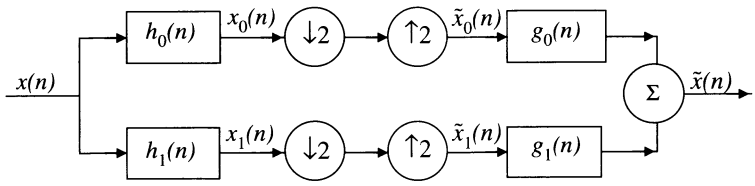

Fig. 3.2 A two-channel filter bank

The PR condition implies to cancel (or at least minimize) three types of distortion, respectively due to the phase and amplitude responses of the filters and to the aliasing that is inevitably introduced by the imperfections of the decimation–interpolation steps. One of the great ideas of sub-band coding, precisely, is that the PR condition does not imply that aliasing should be minimized in each branch separately (which would require close-to-ideal filters) but that some aliasing distortion can be accepted for sub-band signals, *provided all sub-band distortions in adjacent sub-bands*

are cancelled by the final summation. As a consequence, analysis and synthesis filters are not chosen independently.

The combined decimation–interpolation steps, which transform each sub-band signal $x_i(n)$ into $\tilde{x}_i(n)$, replace every other sample in $x_i(n)$ $(i=0,1)$ by zero. It is thus equivalent to adding to each sub-band signal $x_i(n)$ a copy of it in which every other sample is sign-reversed:

$$\tilde{x}_i(n) = \frac{1}{2}\left[x_i(n) + \bar{x}_i(n)\right] \quad with \quad \bar{x}_i(n) = x_i(n)(-1)^n$$
$$\text{or } \tilde{X}_i(z) = \frac{1}{2}\left[X_i(z) + \bar{X}_i(z)\right] \quad with \quad \bar{X}_i(z) = X_i(-z) \tag{3.2}$$

The added signal $\bar{x}_i(n)$ can be seen as a modulation of $x_i(n)$ by $\exp(jn\pi)$, i.e., by a Nyquist rate carrier. Its discrete-time Fourier transform (DTFT) is therefore that of $x_i(n)$, shifted by the Nyquist frequency

$$\bar{X}_i(f) = X_i\left(f - \frac{1}{2}\right) \tag{3.3}$$

which implies that

$$|\bar{X}_i(f)| = |X_i(\frac{1}{2} - f)| \tag{3.4}$$

$|\bar{X}_i(f)|$ is thus the *quadrature mirror* of $|X_i(f)|$, i.e., its mirror with respect to $f = ¼$.

Frequency aliasing occurs in the top branch of Fig. 3.2 when these DTFTs overlap, as can be seen in Fig. 3.3. A similar conclusion could be drawn for the lower branch.

The output of the filter bank can be written as follows:

$$\begin{aligned}\tilde{X}(z) &= \left[\tilde{X}_0(z)G_0(z) + \tilde{X}_1(z)G_1(z)\right] \\ &= \frac{1}{2}\left[X_0(z)G_0(z) + X_1(z)G_1(z)\right] \\ &+ \frac{1}{2}\left[X_0(-z)G_0(z) + X_1(-z)G_1(z)\right]\end{aligned} \tag{3.5}$$

in which the second term is due to the quadrature mirror images of signals $x_0(n)$ and $x_1(n)$ and is therefore responsible for the possible aliasing.

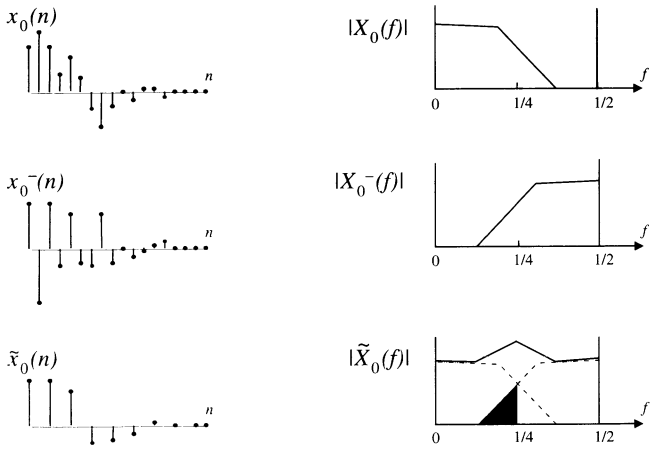

Fig. 3.3 Decimation–interpolation seen as the addition of $x_0^-(n)$ to the low sub-band signal. Due to the overlap of their DTFTs in the black triangle, aliasing occurs, which prevents $|\tilde{X}_0(f)|$ from being equal to $|X_0(f)|$ for f in [0,1/4]

Equation (3.5) can be further expanded as follows:

$$\begin{aligned}\tilde{X}(z) = &\frac{1}{2}X(z)\left[H_0(z)G_0(z)+H_1(z)G_1(z)\right] \\ &+\frac{1}{2}X(-z)\left[H_0(-z)G_0(z)+H_1(-z)G_1(z)\right]\end{aligned} \tag{3.6}$$

which can be identified with (3.1) if

$$\begin{cases} H_0(z)G_0(z)+H_1(z)G_1(z)=2 \\ H_0(-z)G_0(z)+H_1(-z)G_1(z)=0 \end{cases} \tag{3.7}$$

The second condition is easy to satisfy with

$$\begin{aligned} G_0(z) &= H_1(-z) \quad or \quad g_0(n) = h_1^-(n) \\ G_1(z) &= -H_0(-z) \quad or \quad g_1(n) = -h_0^-(n) \end{aligned} \tag{3.8}$$

The first condition in (3.7) then becomes

$$H_0(z)H_1(-z)-H_0(-z)H_1(z)=2 \tag{3.9}$$

QMF filters

One way of satisfying this constraint was introduced by Esteban and Galland (1977), who further imposed $h_1(n)$ to be the *quadrature mirror filter* (QMF) of $h_0(n)$:

$$h_1(n) = h_0^-(n) \quad \text{or} \quad H_1(z) = H_0(-z) \tag{3.10}$$

and found some prototype filter $H_0(z)$ such that

$$\begin{aligned} & H_0{}^2(z) - H_0{}^2(-z) = 2 \\ \text{or} \quad & H_0{}^2(f) - H_0{}^2\left(f - \frac{1}{2}\right) = 2 \end{aligned} \tag{3.11}$$

The final filter bank is given in Fig. 3.4. The beauty of this idea lies in its simplicity.

In practice, $H_0(z)$ is a linear phase (i.e., $h_0(n)$ is symmetrical) FIR filter of length N (N being even[1]), and in order to implement it as a causal filter its impulse response is delayed by $(N-1)/2$. This results in an overall delay of $N-1$ samples in both branches of the filter bank. Taking into account that

$$H_0(f) = |H_0(f)| \, e^{j2\pi f(N-1)/2} \tag{3.12}$$

one finds easily from (3.11)

$$|H_0(f)|^2 + |H_0(f - \frac{1}{2})|^2 = 2 \tag{3.13}$$

It can be shown that designing $H_0(z)$ such that it satisfies (3.13) while being frequency selective is impossible[2]. Johnston (1980) proposed an iterative approach, which results in frequency-selective filter banks free of phase and aliasing distortion, but with some (controllable) amplitude distortion. Many authors have proposed other PR filters with various properties, such as conjugate quadrature filters (CQF). This scientific discussion also led to the parallel development of the *wavelet theory* (see Chapter 10).

---

[1] It can be shown that FIR QMF filters can be built for N odd but require a modification of (3.10).

[2] Satisfying the QMF constraint while not being frequency selective is not very useful in sub-band coding, as it does not allow efficient frequency-dependent SNR control.

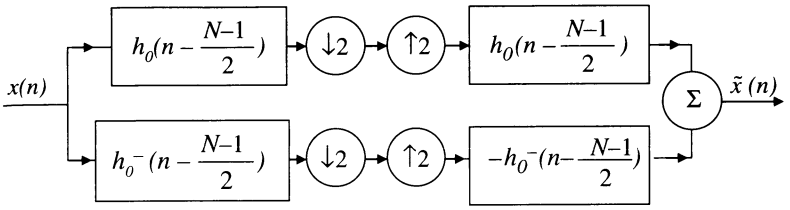

Fig. 3.4 A two-channel QMF filter bank implementing linear phase FIR filters of length N

Pseudo-QMF filters

Going from 2 to M sub-bands can be done by further splitting each sub-band in Fig. 3.4 into two bands and so on (the so-called *tree-structured* or *hierarchical filter bank* approach), but it is more straightforward to implement the filter bank as M parallel passband filters. In the pseudo-QMF (PQMF) approach (Nussbaumer and Vetterli 1984, Vaidyanathan 1993), the analysis filters $H_i(z)$ are obtained by modulating the impulse response $h(n)$ of a low-pass prototype filter with M carrier frequencies f_i uniformly distributed in the range [0,1/2] (Fig. 3.5):

$$\begin{aligned} h_i(n) &= h(n)\cos(2\pi f_i n + \phi_i) \\ g_i(n) &= h(n)\cos(2\pi f_i n + \theta_i) \end{aligned} \quad \text{with } f_i = \frac{(2i+1)}{2M}\frac{1}{2} \tag{3.14}$$

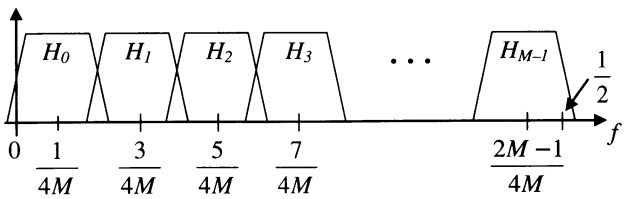

Fig. 3.5 A uniform M-channel filter bank (f denotes normalized frequency)

The values of ϕ_i and θ_i are chosen so that all significant aliased terms are canceled,[3] which can be shown to happen if

[3] The *pseudo* qualifier in PQMF comes from the fact that only adjacent bands are taken into account in the PR condition. Hence PQMF filters do not strictly satisfy the PR condition.

$$\phi_i = -\frac{2i+1}{4}\pi = -\theta_i \tag{3.15}$$

The conditions imposed on the prototype low-pass filter are very similar to Equation (3.13), except that the normalized bandwidth of the filter must be 1/4*M* instead of 1/4. They also cause the analysis and synthesis filters to be the mirror image of each other:

$$h_i(n) = g_i(N-1-n) \tag{3.16}$$

which forces $H_i(z)G_i(z)$ in all sub-band channels to have linear phase.

PQMF filters are used in the MPEG-1 Layer-I audio coding norm.

MDCT filters

PQMF filters were later generalized to PR cosine-modulated filter banks, with special emphasis on filters whose length N is twice the number of channels, known as the modified discrete cosine transform (MDCT) filter banks (Princen and Bradley 1986). The impulse responses of the analysis filters are given by

$$\begin{aligned} h_i(n) &= h(n)\sqrt{\frac{2}{M}}\cos(2\pi f_i n + \phi_i) \\ g_i(n) &= h_i(2M-1-n) \end{aligned} \quad \text{with} \quad \begin{aligned} f_i &= \frac{(2i+1)}{2M}\frac{1}{2} \\ \phi_i &= \frac{(2i+1)(M+1)\pi}{4M} \end{aligned} \tag{3.17}$$

The PR conditions can be shown to lead to the following constraints on the prototype low-pass filter:

$$\begin{aligned} &h^2(n) + h^2(n+M) = 1 \\ &h(n) = h(2M-1-n) \end{aligned} \quad \text{for } n = 0,\ldots,M-1 \tag{3.18}$$

A special case, which is often mentioned, is the modulated lapped transform filter bank (Malvar 1990), in which

$$h(n) = \sin\left[\left(n+\frac{1}{2}\right)\frac{\pi}{2M}\right] \quad \text{for } n = 0,\ldots,2M-1 \tag{3.19}$$

MDCT filters are used in the MPEG-1 Layer-3 (MP3) as well as in the MPEG-2 advanced audio coding (AAC) and MPEG-4 norms.

3.1.2 Filter banks and lapped transforms

If the length of the filters used in M-channel sub-band filters were exactly M samples, one could advantageously implement the operations performed simultaneously by the M analysis filters as a linear transform of successive non-overlapping M sample frames of the input signal.

Conversely, any linear transform can be seen as implementing a special filter bank (Fig. 3.6). If the linear transform is characterized by its $M \times M$ matrix $\mathbf{H}$ and the input frame is stored in column vector $\mathbf{x}$, then $\mathbf{y}=\mathbf{H}\mathbf{x}$ computes the output of M analysis filters whose impulse responses are the time-reversed rows of $\mathbf{H}$. As a matter of fact, Fig. 3.1 verifies

$$\begin{bmatrix} y_0(m) \\ y_1(m) \\ \dots \\ y_{M-1}(m) \end{bmatrix} = \begin{bmatrix} h_0(M-1) & \dots & h_0(1) & h_0(0) \\ h_1(M-1) & \dots & h_1(1) & h_1(0) \\ \dots & \dots & \dots & \dots \\ h_{M-1}(M-1) & \dots & h_{M-1}(1) & h_{M-1}(0) \end{bmatrix} \begin{bmatrix} x(mM-M+1) \\ \dots \\ x(mM-1) \\ x(mM) \end{bmatrix} \tag{3.20}$$

Downsampling is straightforward; since the output of the analysis filters must be downsampled by M, one only needs to compute one value of $\mathbf{y}$ every M samples (by using non-overlapping frames for $\mathbf{x}$). Similarly, computing $\tilde{\mathbf{x}}=\mathbf{G}^T\mathbf{y}$, in which $\mathbf{G}$ is an $M \times M$ matrix whose rows are the time-reversed impulse responses of the analysis filters:

$$\begin{bmatrix} x(mM-M+1) \\ \dots \\ x(mM-1) \\ x(mM) \end{bmatrix} = \begin{bmatrix} g_0(M-1) & g_1(M-1) & \dots & g_{M-1}(M-1) \\ \dots & \dots & \dots & \dots \\ g_0(1) & g_1(1) & \dots & \dots \\ g_0(0) & g_1(0) & \dots & g_{M-1}(0) \end{bmatrix} \begin{bmatrix} y_0(m) \\ y_1(m) \\ \dots \\ y_{M-1}(m) \end{bmatrix} \tag{3.21}$$

computes the output of the synthesis filters, but it also sums all sub-band signals to produce the output frame. Obviously, PR is achieved in this case if $\mathbf{G}^T$ is the inverse of H, since $\mathbf{x}=\mathbf{H}^{-1}\mathbf{y}$.

Having input and output frames that do not overlap is a problem when sub-band signals are modified in a time-varying way (as it happens typically when they are quantized as a function of their instantaneous energy); the effect of each modification applies to one synthesis frame only, with no possibility of interframe smoothing. This effect is known as

the *blocking effect*.[4] Hence practical filter banks make use of longer analysis and synthesis filters (as in the previous section on PQMF filters).

However, when the length N of the analysis and synthesis filters is longer than the number of channels, the filtering operations can still be implemented as the multiplication of N sample frames with $N \times M$ or $M \times N$ matrices. Analysis filtering is still achieved by $\mathbf{y}=\mathbf{Hx}$, in which $\mathbf{H}$ is an $M \times N$ matrix whose rows are the time-reversed impulse responses of the analysis filters and $\mathbf{x}$ is an N sample frame. Downsampling is achieved by shifting successive input frames by M samples (i.e., they *overlap* by N–M samples).

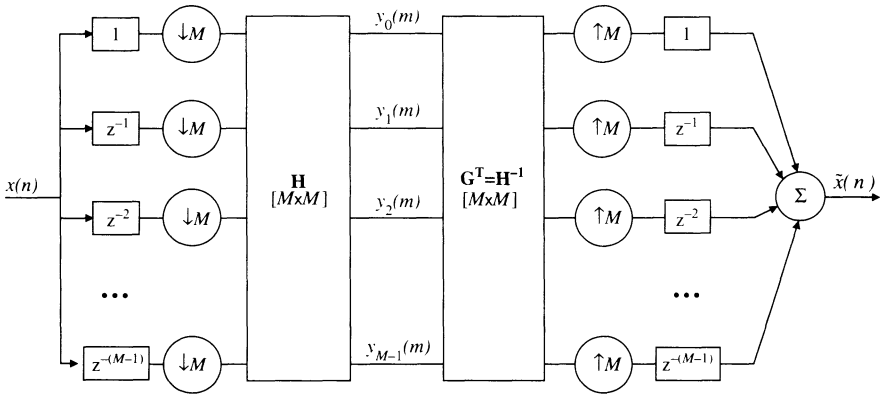

Fig. 3.6 An M-channel transform-based implementation of a filter bank using filters of length N=M

The upsampling, synthesis filtering, and summation operations are still performed as $\mathbf{G}^T\mathbf{y}$, in which $\mathbf{G}$ is an $M \times N$ matrix whose rows are the (not time-reversed) impulse responses of the analysis filters. This computes the sum of the responses of the synthesis filters each excited by the single sample stored in $\mathbf{y}$ for each sub-band. Just as the input frames, the output frames overlap in time (and should therefore be accumulated; see Fig. 3.7 for a simple example with $N = 2M$). This leads to the more general implementation of Fig. 3.8, which differs only slightly from Fig. 3.6 and still clearly shows the underlying sub-band processing interpretation.

[4] The same concept will be for the case of image coding in Chapter 8.

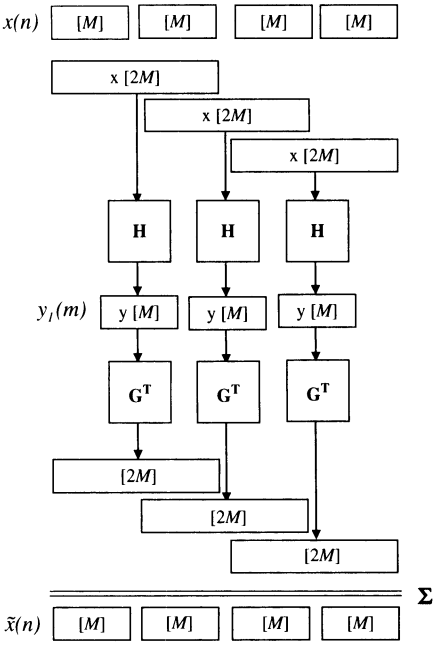

Fig. 3.7 Frame-based view of an M-channel transform-based implementation of a filter bank using filters of length $N=2M$

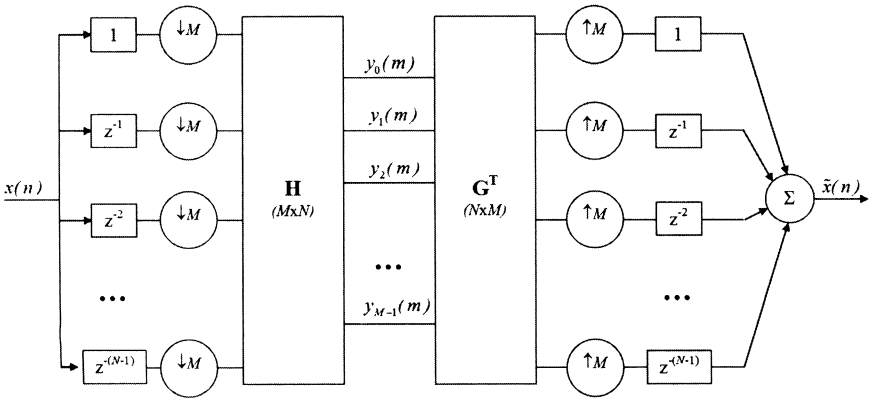

Fig. 3.8 An M-channel transform-based implementation of a filter bank using filters of length $N>M$

3.1.3 Using the masking properties of the human ear

From an engineering point of view, the human ear (not only the mechanical parts of it but also the whole neural system that produces an acoustic sensation) is a complex device in many respects (Painter and Spanias 2000). First, its sensitivity depends on frequency: to be just heard, a narrow-band stimulus must have a sound pressure level higher than a threshold called the *absolute auditory threshold,* which is a function of frequency (Fig. 3.9, left). Second, for a given frequency, the impression of loudness is not linearly related to sound pressure (this is obvious for the absolute auditory threshold but can be generalized for all loudness values). Third, the sensitivity of the ear to a given stimulus is not the same if this stimulus is presented alone or if it is mixed with some masking stimulus. This is termed as the *masking effect* and is approximated as a local modification of the auditory threshold in a frequency band surrounding that of the masking stimulus (Fig. 3.9, right). Even worse, this effect depends on the type of masking stimulus (pure tones and narrow-band noise do not have the same masking effect). Last but not least, not everyone is equal with respect to sound: the acoustic sensation is listener-dependent and varies with time.

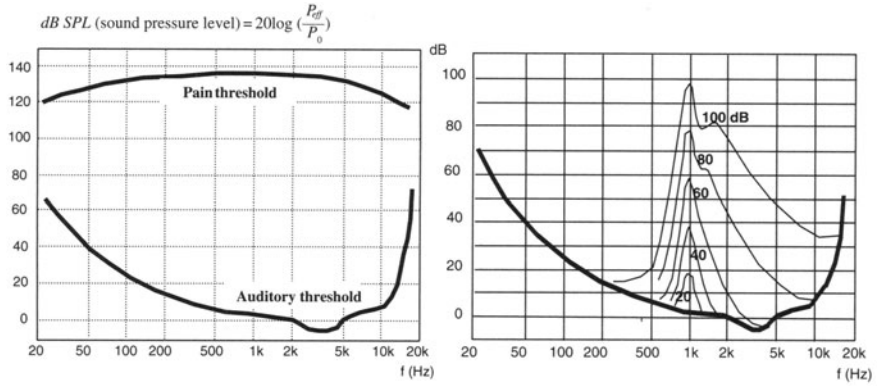

Fig. 3.9 *Left*: Auditory and pain thresholds as a function of frequency. *Right*: Masking effect due to a pure tone at 1,000 Hz for various intensities. Sounds below the modified auditory threshold are "hidden" by the masking tone. A pure tone at 1,000 Hz with an intensity of 80 dB SPL,[5] for instance, will make another pure tone at 2,000 Hz and 40 dB SPL inaudible

[5] Sound pressure level (SPL) L_p is a logarithmic measure of the root-mean-square sound pressure of a sound relative to a reference value, usually measured in decibel:

One of the most practical features of the masking effect can be modeled as operating within a discrete set of 25 auditory "channels" termed as *critical bands* (with idealized band edges equal to 20, 100, 200, 300, 400, 510, 630, 770, 920, 1,080, 1,270, 1,480, 1,720, 2,000, 2,320, 2,700, 3,150, 3,700, 4,400, 5,300, 6,400, 7,700, 9,500, 12,000, and 15,500 Hz). As an example, if a signal and a masker are presented simultaneously, then only the masker frequencies falling within the critical bandwidth contribute to masking the signal. In other words, the ear acts as a sub-band processing system, in which separate nonlinear masking effects mainly appear in separate sub-bands.

The great interest of sub-band coding is to establish a master–slave relationship (Fig. 3.10), in each sub-band, between (i) the local *masking threshold* computed by the so-called *psychoacoustic model* from an estimation of the masking stimuli present in the auditory spectrum and (ii) quantization noise, i.e., the number of bits used to quantize the corresponding sub-band signal.

Note that although sub-band coders are based on the features of the auditory process, they do not completely mimic it. As an example, although the edges of the critical bands are not uniformly spaced in frequency, most sub-band coders use equal bandwidth channels. More importantly, since the masking effect depends on the actual loudness of the masking signal, while the coder is not aware of the actual volume level set by the listener, the psychoacoustic model used in sub-band coding a priori assumes that the user sets the volume level such that the least significant bit of the original 16-bit PCM code can just be heard.

3.1.4 Audio coders

MPEG-1 audio

The MPEG-1 audio norm actually defines three complexity layers of processing, each with its own sub-band coding scheme, psychoacoustic model, and quantizer (see Pan 1995 or Painter and Spanias 2000 for a tutorial). Starting from 1.4 Mbps for a stereo audio signal, Layer-I achieves a 1:4 compression level (down to 384 kbps). With the addition of Layer-II (formerly known as Musicam: Masking Pattern adapted Universal Subband Coding And Multiplexing), a 1:6–8 ratio is reached (between 256 and 192 kbps). This coder is very much used in digital video broadcasting (DVB). MPEG-1 Layer-III, commercially known as MP3,

$$L_p = 20\log_{10}\left(p_{rms} / p_{ref}\right)$$

where p_{ref} is the reference sound pressure, usually set to 20 μPa (rms), although 1 μPa (rms) is used for underwater acoustics.

further compresses the data stream down to 128–112 kbps (i.e., a compression ratio of 1:10–12). In this chapter, we examine only Layer-I, which is far enough for a proof of concept.

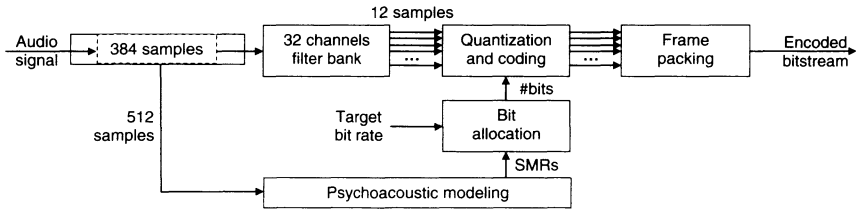

Fig. 3.10 Psychoacoustic model controlling sub-band quantizers in the MPEG-1 Layer-I audio coder

The MPEG-1 Layer-I coder (Fig. 3.10) implements a 32-channel sub-band coder. Samples are processed by frames of 384 samples (i.e., 12 samples at the downsampled rate). The psychoacoustic model is based on a 512-sample FFT estimation of the local power spectral density of the signal, $S_{xx}(f)$. Local maxima with dominant peaks are detected as tonal maskers and extracted from the FFT. A single noise masker is then computed from the remaining energy in each critical band. Maskers are then combined, accounting again for the existence of critical bands (typically, several maskers in a single band are merged into the strongest of them). Individual masking thresholds are finally estimated for each tonal and noise masker and summed to produce the overall masking threshold, $\Phi(f)$. This leads to the determination of a *signal-to-mask ratio* (SMR) in each band:

$$SMR(k) = 10\log_{10}\left(\frac{\max\limits_{\text{band } k} S_{xx}(f)}{\min\limits_{\text{band } k} \Phi(f)}\right) \tag{3.22}$$

Sub-band signals are quantized uniformly, and the number of bits for each sub-band is chosen so as to make resulting signal-to-noise ratio lower than the SMR. Higher-frequency sub-bands typically require less bits. In particular, sub-bands 27–32 are assigned 0 bits by design. Quantization is made adaptive by normalizing sub-band signals before quantizing them. Normalization is performed by blocks of 12 sub-band samples (i.e., by blocks of 384 samples at the original sampling frequency) by estimating a local scale factor and dividing samples by its value (Fig. 3.11).

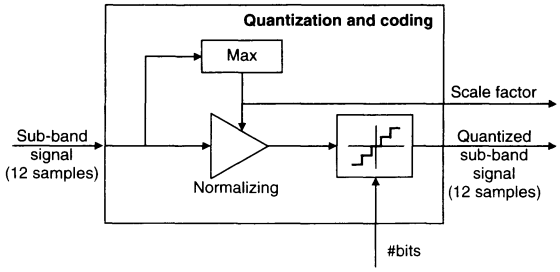

Fig. 3.11 Adaptive quantization in the MPEG-1 Layer-I audio coder

MPEG-1 Layer-II is based on the same principle but uses 1024 samples for FFT computation in its psychoacoustic model and processes frames that are three times longer than in Layer-I.[6] It also implements nonsimultaneous (sometimes called "temporal") masking, which denotes the fact that masking can occur both before masker onset and after masker removal.

MPEG-1 Layer-III further splits each of the 32 original sub-bands into 18 channels by using a modified discrete cosine transform (MDCT, see Section 3.1.2), which provides a better frequency resolution[7] when required (i.e., in stationary parts of the audio signal). Its coding scheme is dynamic; more bits are allocated to frames that require it.

MPEG-2 Audio – Advanced audio coding

MPEG-2 was normalized in 1994. The MPEG-2 BC audio coder adds a series of features to the MPEG-1 audio coder: an extension to lower sampling frequencies (down to 8 kHz) and multichannel processing for 5.1 surround sound. It remains *backward compatible* (BC) with the MPEG-1 coder, i.e., MPEG-1 audio coders can decode MPEG-2 BC, ignoring the additional information.

MPEG-2 AAC (advanced audio coding[8]) appeared in 1997 and provides a great improvement over MPEG-2 BC (hence, over MP3), especially for low bit rates (below 100 kbps). Contrary to the MPEG-1 Layer-III coder,

---

[6] More specifically, scale factors are computed as before but quantized by groups of three.

[7] MPEG-1 Layers I and II have a good time resolution, since sub-band samples are updated every 32 audio samples, which enables them to track audio transients. They exhibit a bad frequency resolution though, as their sub-band filter bandwidth is $F_s/2/32 \approx 700$ Hz.

[8] Not to be confused with AC3, the Dolby Digital 5.1 audio coding scheme developed by Dolby Labs.

which added an MDCT filter bank for increasing the frequency resolution when needed, this coder uses a single filter bank, with adaptive block size: large 1024-sample frames (resp., short 128-sample frames) are used for stationary (resp., transient) sounds, which provide a good compromise between time and frequency resolution. This coder provides 5.1 transparency at 384 kbps. It is the most commonly used format for compressing audio CDs for Apple's iPod and iTunes.

MPEG-2 audio coders are also used in digital TV broadcasting.

MPEG-4 audio

The AAC coder has been enhanced in the framework of MPEG-4 by including additional tools (such as long-term prediction) for intermediate bit rates. A low-delay version of the coder is also available for teleconference applications. The resulting MPEG-4 AAC coder, normalized in 1999, provides stereo transparency with a bit rate between 96 and 128 kbps and 5.1 transparency at 256 kbps.

In 2003, a technique known as *sub-band replication* (SBR) has been added to the MPEG-4 AAC coder, leading to the MPEG-4 HE AAC (high efficiency). The idea is to use the strong correlation between low-frequency (up to 4–8 KHz) and higher frequency spectral components of audio signals and to avoid sending redundant information. HE AAC provides stereo transparency at 48 kbps and 5.1 transparency at 128 kbps.

A new version of it, HE AAC v2 (sometimes called AAC+), appeared in 2004, which further uses psychoacoustic effects to better encode stereo channels as one main channel and side information for spatialization. Intermediate stereo quality is obtained at 24 kbps. This coder is used in the emerging digital radio mondiale and in 3G mobile telephony (the 3GPP standard).

The latest development has been reached in July 2006 with the definition of the HE AAC surround (or MPEG surround), which adapts the spatialization coding principles of HE AAC v2 to multichannel audio. As a result, 5.1 transparency is now available at 64 kbps.

3.2 MATLAB proof of concept: ASP_mp3.m

In this section, we provide a proof of concept of the MPEG-1Layer-I audio coding norm. We start by examining a two-channel filter bank using conventional filters (Section 3.2.1) and QMF filters (Section 3.2.2). We then extend our study to a 32-channel PQMF filter bank (Section 3.2.3)

and show how it can be efficiently implemented using block-based lapped transforms (Section 3.2.4). We conclude by providing our filter bank with a perceptual quantizer for sub-band signals (Section 3.2.5).

3.2.1 Two-channel filter bank

Before examining a complete M-channel filter bank, we first discuss the two-channel case and the effect of a decimation and interpolation in each branch.

Let us first generate a 4-seconds chirp signal (from 0 to 4 kHz) with a sampling rate of 8 kHz, which we will use as a reference throughout this section, and plot its spectrogram (Fig. 3.12, left).

```
Fs=8000;
input_signal=chirp((0:4*Fs)/Fs,0,4,4000);
specgram(input_signal,1024,Fs,256);
```

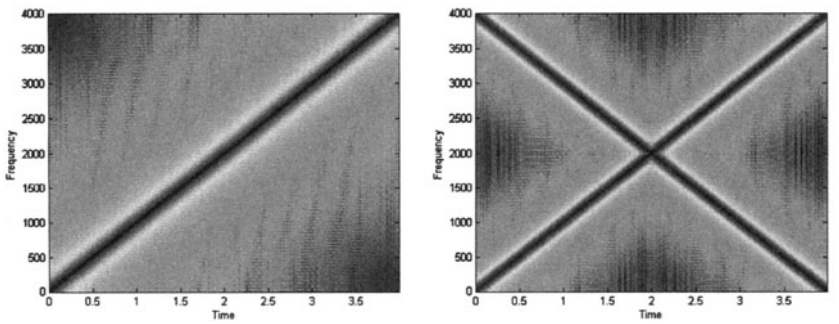

Fig. 3.12 Spectrogram of a chirp. *Left*: Original signal; *right*: after downsampling–upsampling without using analysis and synthesis filters

Applying direct downsampling–upsampling by 2 results in replacing every other sample by zero. The result is that an artifact signal is added, whose spectrum is the quadrature mirror image of that of the chirp. Two sinusoids can be heard at any time (Fig. 3.12, right).[9]

```
downsampled=input_signal(1:2:end);
upsampled(1:2:2*length(downsampled))=2*downsampled;
```

[9] Note that we multiply the signal by 2 when upsampling, so that the power of the signal remains unchanged.

Adding a quarter-band filter after upsampling eliminates the image distortion, making sure that only one sinusoid appears at any time (Fig. 3.13). During the first half of the chirp, that sinusoid is the chirp itself; during the second half, the sinusoid is an alias due to the quadrature mirror image of the chirp. We design the synthesis filter as a symmetric FIR, so as to avoid phase distortion, and we set its order to 1,000, so as to obtain high stop-band rejection (close to 80 dB). Note that the first 1,000 samples are a transient.

```
G0=fir1(1000,1/2);
freqz(G0,1);
G0_output=filter(G0,1,upsampled);

specgram(G0_output(1001:end),1024,Fs);
```

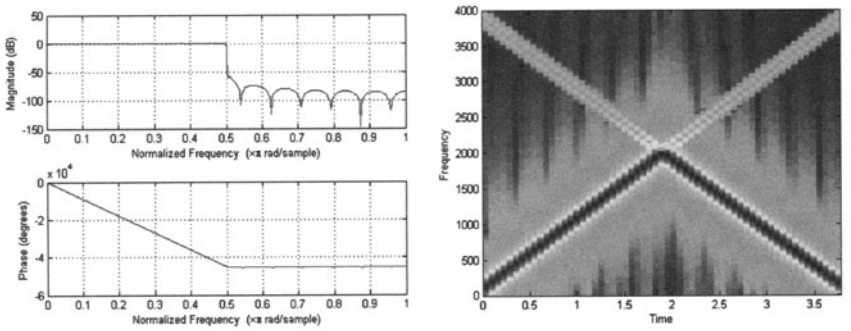

Fig. 3.13 *Left*: Frequency response of the quarter-band analysis filter; *right*: spectrogram of a chirp after downsampling–upsampling using a low-pass synthesis filter only

Adding another quarter-band filter before downsampling removes the aliasing distortion, i.e., the alias in the second half of the chirp (Fig. 3.14, left). This synthesis filter can a priori be identical to the analysis filter.

```
H0=G0;
H0_output=filter(H0,1,input_signal);
downsampled=H0_output(1:2:end);
upsampled(1:2:2*length(downsampled))=2*downsampled;
G0_output=filter(G0,1,upsampled);

specgram(G0_output(2001:end),1024,Fs, 256);
```

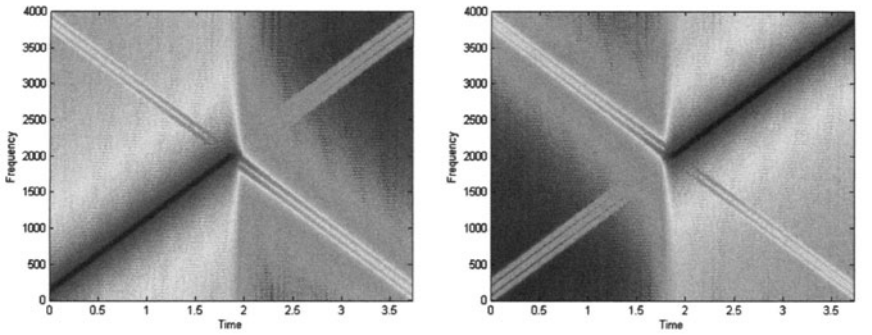

Fig. 3.14 *Left*: Spectrogram of a chirp after downsampling–upsampling using low-pass analysis and synthesis filters; *right*: same thing with high-pass analysis and synthesis filters

Using HF quarter-band filters instead of LF ones selects the second half of the chirp (Fig. 3.14, right).

```
G1=fir1(1000,1/2,'high');
H1=G1;
H1_output=filter(H1,1,input_signal);
downsampled=H1_output(1:2:end);
upsampled(1:2:2*length(downsampled))=2*downsampled;
G1_output=filter(G1,1,upsampled);
```

We now examine the output of the two-channel sub-band filter bank by adding the two signals obtained above (Fig. 3.15, left). Perfect reconstruction is not achieved, as shown in the previous spectrogram. Closer examination of the error waveform (Fig. 3.15, right) shows that most of the error lies in the center of the signal (this is not shown in the spectrogram), i.e., for frequencies for which the `H0` and `H1` filters overlap; more important, aliasing cannot be avoided in each band for these frequencies.

```
synt_signal=G0_output+G1_output;
% Shift synt_signal to account for the filters delay.
% A symmetric FIR filter of order N (length N+1) brings
% a delay of N/2 samples.
error=synt_signal(1001:end)-input_signal(1:end-1000);
```

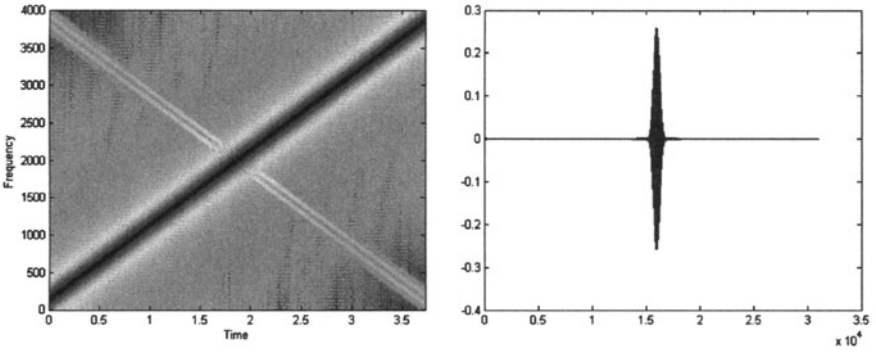

Fig. 3.15 *Left*: Spectrogram of the output of the two-channel filter bank; *right*: error signal waveform

3.2.2 Two-channel QMF filter bank

Perfect reconstruction is possible even with nonideal filters (i.e., even with some aliasing in each band) provided the overall aliasing is canceled when adding the low-pass and high-pass sub-band signals. This, as we shall see, is mainly the responsibility of the analysis and synthesis filters. Johnson's type QMF filters provide a solution for nearly perfect reconstruction. In this example, we use Johnson's "B-12" QMF (Fig. 3.16).

```
H0_QMF=[-0.006443977 0.02745539 -0.00758164 ...
    -0.0913825  0.09808522 0.4807962];
H0_QMF=[H0_QMF fliplr(H0_QMF)];
[H0,W]=freqz(H0_QMF,1);

H1_QMF=H0_QMF.*[1 -1 1 -1 1 -1 1 -1 1 -1 1 -1];
[H1,W]=freqz(H1_QMF,1);
```

These filters are not very frequency selective (because their order is low) and will therefore allow important aliasing in each band. This is confirmed by passing our chirp signal through the corresponding filter bank (Fig. 3.17).

```
% LF band
H0_output=filter(H0_QMF,1,input_signal);
subband_0=H0_output(1:2:end);
upsampled(1:2:2*length(subband_0))=2*subband_0;
G0_QMF=H0_QMF;
G0_output=filter(G0_QMF,1,upsampled);

% HF band
H1_output=filter(H1_QMF,1,input_signal);
subband_1=H1_output(1:2:end);
upsampled(1:2:2*length(subband_0))=2*subband_1;

G1_QMF=-H1_QMF;
```

```
G1_output=filter(G1_QMF,1,upsampled);
synt_signal=G0_output+G1_output;
```

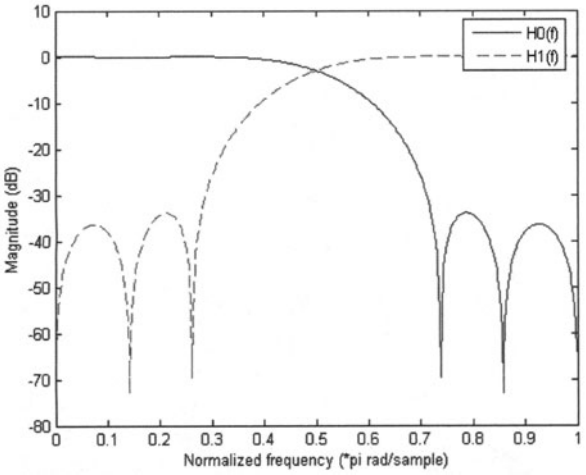

Fig. 3.16 *Left*: Spectrogram of the output of the two-channel filter bank; *right*: error signal waveform

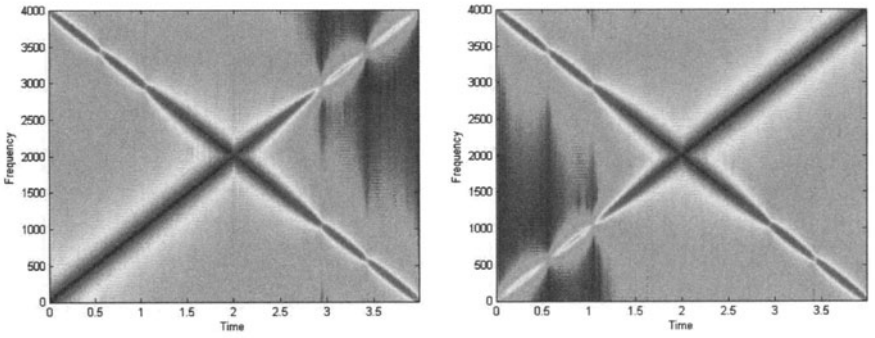

Fig. 3.17 *Left*: Spectrogram of the output of the LF channel of the QMF filter bank; *right*: spectrogram of the output of the HF channel. Important aliasing can be observed

However, perfect reconstruction is now achieved, because the QMF analysis and synthesis filters are such that the aliasing in each band sum up to zero[10] (Fig. 3.18).

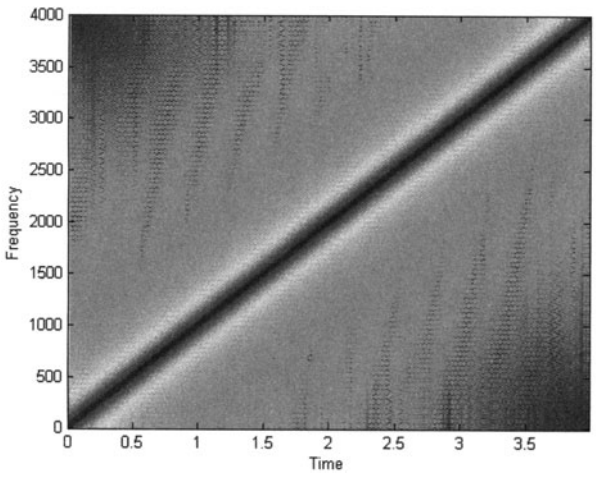

Fig. 3.18 Spectrogram of the output of the two-channel QMF filter bank

3.2.3 32-channel pseudo-QMF filter bank

We now build a 32-channel PQMF filter bank, as implemented in the MPEG-1 Layer-I norm, and check its perfect reconstruction capability.

MATLAB function involved:

- `hn = PQMF32_prototype` returns in `hn` the impulse response of the prototype low-pass symmetric filter of length 512 for building a 32-channel PQMF filter bank. This filter is used in the MPEG-1 Layer-I coder. Its bandpass is $F_s/2/32/2$=344.5 Hz and it satisfies the PR condition.

We first build the filter bank from the prototype filter and examine the magnitude frequency responses of all filters (Fig. 3.19, left). Clearly, PQMF filters are not ideal bandpass filters with bandwidth=689 Hz, but their response overlaps only with their two neighboring filters (hence their name of "pseudo"-QMF filters). The total bandwidth of each filter is

[10] More precisely, very close to 0.

indeed about 2 × 689 Hz. Note that the filter gain is 15 dB, i.e., 20*log10(32)/2. As a result, passing twice through the filter produces a gain of 32, which compensates for the decimation by 32. In the next lines, it is thus not necessary to multiply upsampled sub-band signals by 32.

```
% Load the prototype lowpass filter
hn=PQMF32_prototype;

% Build 32 cosine modulated filters centered on normalized
% frequencies Fi=(2*i+1)/64 *1/2
PQMF32_Gfilters = zeros(32, 512);
for i = 0:31
    t2 = ((2*i+1)*pi/(2*32))*((0:511)+16);
    PQMF32_Gfilters(i+1,:) = hn.*cos(t2);
end
PQMF32_Hfilters=fliplr(PQMF32_Gfilters);

for i = 1:32
    [H,W]=freqz(PQMF32_Hfilters(i,:),1,512,44100);
    plot(W,20*log10(abs(H))); hold on
end
xlabel('Frequency (Hz)'); ylabel('Magnitude (dB)');
set(gca,'xlim',[0 22050]);
ylabel('Magnitude (dB)');
hold off;
```

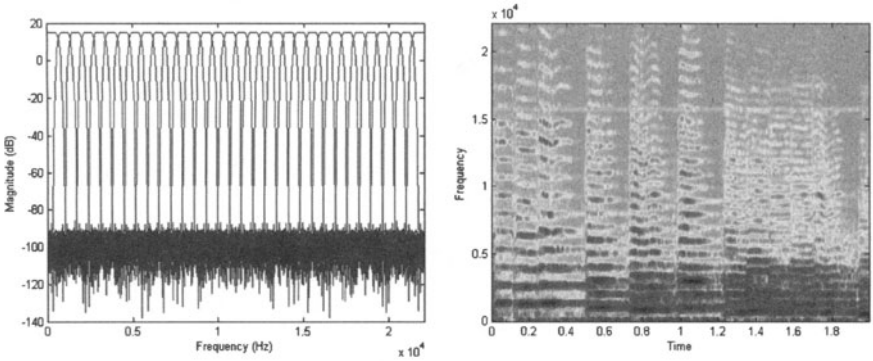

Fig. 3.19 *Left*: Magnitude frequency response of the 32 analysis filters of a PQMF filter bank; *right*: spectrogram of an audio signal containing a few notes played on the violin

Let us now check the output of the PQMF filter bank when fed with 2 seconds of violin monophonic signal sampled at 44.100 Hz (Fig. 3.19, right). As revealed by listening to sub-band 3, isolated sub-band signals are very much aliased, because each PQMF filter is not ideal (Fig. 3.20, left).

```
[input_signal,Fs]=wavread('violin.wav');

output_signal=zeros(size(input_signal));
```

```
for i=1:32

    Hi_output=filter(PQMF32_Hfilters(i,:),1,input_signal);
    subband_i=Hi_output(1:32:end);
    upsampled_i(1:32:32*length(subband_i))=subband_i;

    % synthesis filters are the symmetric of the analysis
    % filters, which ensures linear phase in each sub-band
    Gi_output=filter(PQMF32_Gfilters(i,:),1,upsampled_i');
    output_signal=output_signal+Gi_output;

    if i==3
        G3_output=Gi_output;
    end;

end;
```

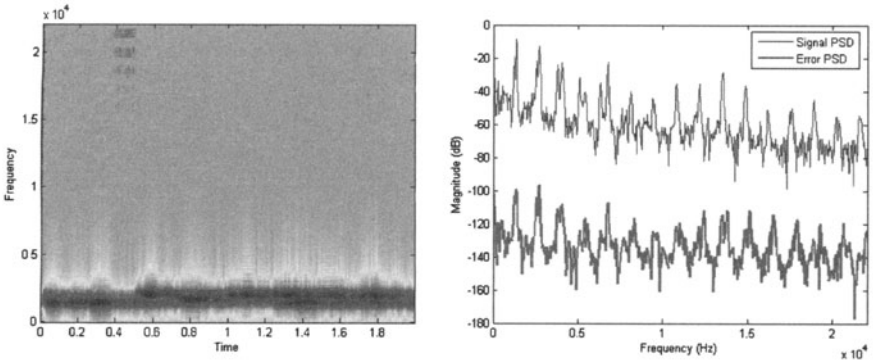

Fig. 3.20 *Left*: Spectrogram of sub-band 3; *Right*: periodogram of the original and error signals, around sample 11,000.

The PQMF filter bank, however, makes sure that aliasing in adjacent bands cancels itself when sub-bands are added. The power of the reconstruction error is about 85 dB below that of the signal (Fig. 3.20, right). Note that the output is delayed by 511 samples (since analysis and synthesis filters have a delay of 511/2 samples).

MATLAB function involved:

- `snr=snr(signal,signal_plus_noise,max_shift,showplot)` returns the signal-to-noise ratio computed from the input signals. `Max_shift` gives the maximum time shift (in samples) between `signal` and `signal_plus_noise`. The actual time shift (obtained from the maximum of the cross-correlation between both signals) is taken into account to estimate the noise. If `showplot` is specified, then the signal,

signal_plus_noise, and error are plotted, and the SNR is printed on the plot.

```
error=output_signal(512:end)-input_signal(1:end-511);

[signal_psd,w]=periodogram(input_signal(11001:12024),...
                    hamming(1024));
[error_psd,w]=periodogram(error(11001:12024),...
                    hamming(1024));
plot(w/pi*22050,10*log10(signal_psd));
hold on;
plot(w/pi*22050,10*log10(error_psd),'r','linewidth',2);
hold off;

snr_PQMF=snr(input_signal(1:end-511),output_signal(512:end),0)
```

```
snr_PQMF =  84.2443
```

3.2.4 Filter banks and lapped transforms

We now show how analysis and synthesis filters can be implemented using block transforms.

When the length of the filters used in M-channel sub-band filters are exactly M samples, these operations reduce to a linear transform of successive non-overlapping M sample frames of the input signal. A four-sample DFT, for instance, can implement a four-channel filter bank whose sub-band filter impulse responses are the time-reversed of the lines of the 4×4 DFT matrix. Applying it to our chirp is straightforward.

```
Fs=8000;
input_signal=chirp((1:4*Fs)/Fs,0,4,4000);

for i=1:length(input_signal)/4

    % creating a column vector with 4 samples
    input_frame=input_signal(4*(i-1)+1:4*(i-1)+4)';

    % producing one sample in each downsampled sub-band, i.e.
    % band-pass filtering and downsampling all sub-bands in
    % one operation.
    subbands=fft(input_frame);

    % producing four samples of the filter bank output, i.e.
    % upsampling, band-pass filtering all sub-bands, and
    % summing them in one operation.
    output_frame=ifft(subbands);

    % storing the output column vector in the output signal
    output_signal(4*(i-1)+1:4*(i-1)+4)=output_frame';

end;

%soundsc(output_signal,Fs);
```

Obviously, this will return the original chirp. Since the underlying filters have complex coefficients, however, each sub-band signal is complex. Moreover, this type of filter bank is not very frequency selective, as shown in Fig. 3.21. The frequency overlap between adjacent bands is about half the main lobe bandpass (as in the previous section on PQMF), but the side lobes are very high. This does not make it a good candidate for sub-band coding.

```
tmp=[1 ; exp(-j*pi/2) ; exp(-j*pi) ; exp(-j*3*pi/2)];
DFT_matrix_4x4=vander(tmp);

for i=1:4
    [H,W]=freqz(fliplr(DFT_matrix_4x4(i,:)),1,'whole');
    plot(W/pi,max(20*log10(abs(H)),-50)); hold on
end
```

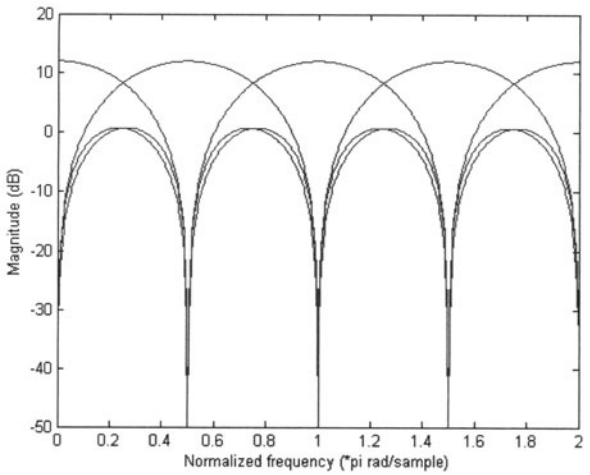

Fig. 3.21 Magnitude frequency response of the four channels of a DFT filter bank. Note that we show the responses in the full [0,1] frequency range, as these frequency responses are not two-sided

In general, the length of the impulse responses of the analysis and synthesis filters used in sub-band coders is higher than the number M of channels. The filtering operations, however, can still be implemented as the multiplication of L-sample frames with $L \times M$ or $M \times L$ matrices. For example, the 32-channel PQMF filter bank introduced in the previous section, in which the length N of the impulse response of each filter is 512 samples, can be efficiently implemented as follows (which is very much similar to the implementation of our previous DFT-based filter bank, with

the addition of overlap). Obviously, we get the same results as before (Fig. 3.22).

```
[input_signal,Fs]=wavread('violin.wav');

% Block-based sub-band filtering
input_frame=zeros(512,1);
output_signal=zeros(size(input_signal));

for i=1:(length(input_signal)-512+32)/32

    % Overlap input_frames (column vectors)
    input_frame=input_signal((i-1)*32+1:(i-1)*32+512);

    % Analysis filters and downsampling
    % Since PQMF H filters are the time-reversed G filters,
    % we use the G filters matrix to simulate analysis
    % filtering
    subbands_frame_i = PQMF32_Gfilters*input_frame;

    % Synthesis filters
    output_frame = PQMF32_Gfilters'*subbands_frame_i;

    % Overlap output_frames (with delay of 511 samples)
    output_signal((i-1)*32+1:(i-1)*32+512)= ...
        output_signal((i-1)*32+1:(i-1)*32+512)+output_frame;

end

error=output_signal(512:end)-input_signal(1:end-511);
```

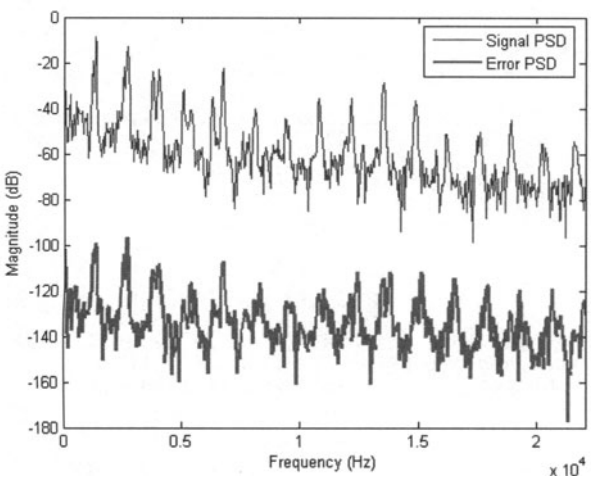

Fig. 3.22 Periodogram of the original and error signals, around sample 11,000

3.2.5 Perceptual audio coding

The sub-band filtering process developed in the previous sections transforms the original stream of samples at sampling frequency F_s into 32 parallel sub-bands sampled at $F_s/32$. It still needs a quantization and coding step to produce overall bit rate compression.

Quantizing sub-band samples uniformly[11] does not allow much transparency at low bit rates, as shown in the four-bits per sub-band sample trial below (compression factor = 4). The resulting output signal is degraded; its spectrogram is given in Fig. 3.23.

```
% Build the PQMF H and G filters
hn=PQMF32_prototype;
PQMF32_Gfilters = zeros(32, 512);
for i = 0:31
    t2 = ((2*i+1)*pi/(2*32))*((0:511)+16);
    PQMF32_Gfilters(i+1,:) = hn.*cos(t2);
end

[input_signal,Fs]=wavread('violin.wav');

% Block-based sub-band analysis filtering
input_frame=zeros(512,1);
output_signal=zeros(size(input_signal));

n_frames=(length(input_signal)-512+32)/32;
for i=1:n_frames

     % Overlap input_frames (column vectors)
     input_frame=input_signal((i-1)*32+1:(i-1)*32+512);

     % Analysis filters and downsampling
     % NB: we put sub-band signals in columns
     subbands(i,:) = (PQMF32_Gfilters*input_frame)';

     % Uniform quantization on 4 bits, using a mid-thread
       quantizer in [-1,+1]
      n_bits = 4;
      alpha = 2^(n_bits-1);
      quantized_subbands(i,:) =…
        (floor(alpha*subbands(i,:)+0.5))/alpha; % mid-thread

     % Synthesis filters
     output_frame = PQMF32_Gfilters'*quantized_subbands(i,:)';

     % Overlap output_frames (with delay of 511 samples)
     output_signal((i-1)*32+1:(i-1)*32+512)= ...
         output_signal((i-1)*32+1:(i-1)*32+512)+output_frame;

end
```

[11] We use a *mid-thread* quantizer here, as opposed to the more classical *mid-rise* quantizer (see Section 2.1.1), so as to encode low-level signals to 0. Failing to do so quickly creates *musical noise*, caused by high-frequency changes in the LSB for low-level signals. Note that we do not use the uencode and udecode functions from MATLAB, which do not implement a true mid-thread quantizer.

The output signal is strongly distorted. The SNR falls down to 10.3 dB.

```
% NB: no delay compensation required, as the first sub-band
% samples produced by the lapped transform correspond to the
% 512th original samples.

error=output_signal-input_signal;

[signal_psd,w]=periodogram(input_signal(11001:12024),...
    hamming(1024));
[error_psd,w]=periodogram(error(11001:12024),...
    hamming(1024));
plot(w/pi*22050,10*log10(signal_psd));
hold on;
plot(w/pi*22050,10*log10(error_psd),'r','linewidth',2);
hold off;

snr_4bits=snr(input_signal(512:end-512),...
              output_signal(512:end-512),0)
```

```
snr_4bits = 10.3338
```

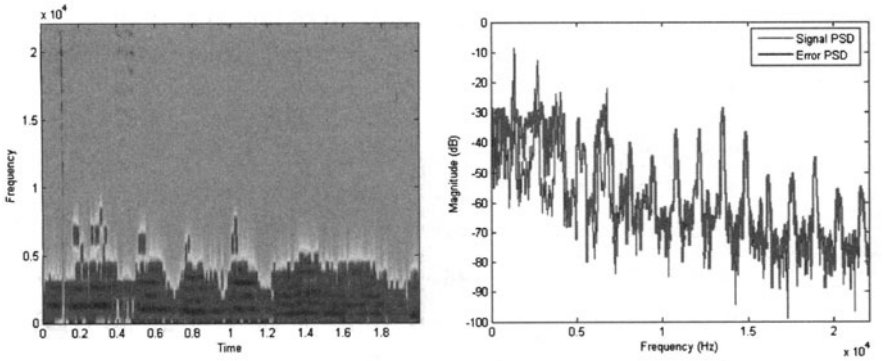

Fig. 3.23 Sub-band coding using fixed four-bit uniform quantization. *Left*: spectrogram; *right*: the periodograms of the original and error signals (computed here around sample 11,000) are almost identical, except for low frequencies

One can see that this fixed [–1,+1] quantizer range does not adequately account for the variation of sub-band signal level across sub-bands, as well as in time (Fig. 3.24, left).

```
plot(quantized_subbands(100:200,2)); hold on;
plot(subbands(100:200,2),'--r'); hold off;
```

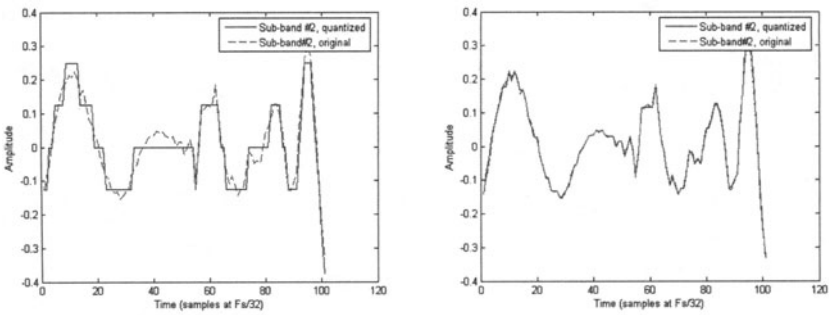

Fig. 3.24 Excerpt from sub-band #2 before and after quantization. *Left*: using fixed four-bit uniform quantization; *right*: using level-adaptive four-bit uniform quantization

An obvious means of enhancing its quality is therefore to apply a scale factor to each sub-band quantizer. As in the MPEG-1 Layer-I coder, we compute a new scale factor every 12 sub-band sample (i.e., every 32×12 = 384 sample at the original sample rate). The resulting quantization errors are much reduced (see Fig. 3.24, right).

```
% Adaptive quantization per blocks of 12 frames
n_frames=fix(n_frames/12)*12;
for k=1:12:n_frames

    % Computing scale factors in each 12 samples sub-band
    % chunk
    [scale_factors,tmp]=max(abs(subbands(k:k+11,:)));

    % Adaptive uniform quantization on 4 bits, using a
    % mid-thread quantizer in [-Max,+Max]
    for j=1:32 % for each sub-band

        n_bits = 4;
        alpha = 2^(n_bits-1)/scale_factors(j);
        quantized_subbands(k:k+11,j) = ... % mid-thread
            (floor(alpha*subbands(k:k+11,j)+0.5))/alpha;

    end;

end;
```

The resulting signal is now of much higher quality, as shown in Fig. 3.25. The overall SNR has increased to 25 dB.

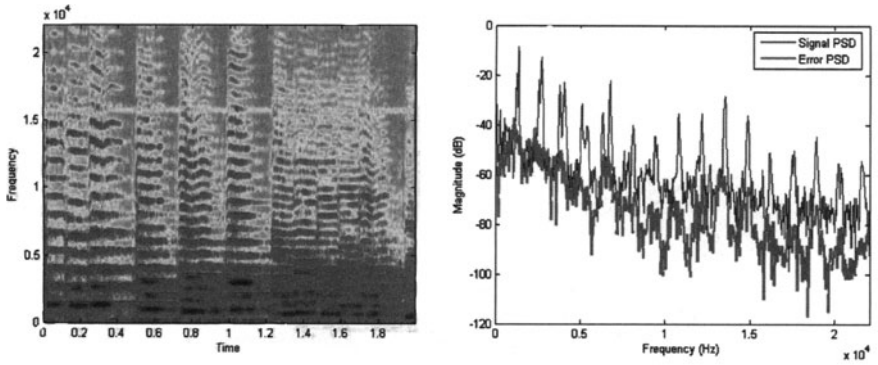

Fig. 3.25 Sub-band coding using adaptive four-bit uniform quantization. *Left*: spectrogram; *right*: periodogram of the original and error signals, around sample 11,000.

```
snr_4bits_scaled=snr(input_signal(512:end-512),…
                     output_signal(512:end-512),0)
```

```
snr_4bits_scaled =   25.0629
```

The ultimate refinement, which is by far the most effective and results from years of audio research, consists in accepting more quantization noise (by allocating less bits and thereby accepting a lower SNR) in frequency bands where it will not be heard and using these extra bits for more perceptually prominent bands. The required perceptual information is provided by a psychoacoustic model.

For any 512-sample frame taken from the input signal, the MPEG-1 Layer-I psychoacoustic model computes a global masking threshold, obtained by first detecting prominent tonal and noise maskers separately and combining their individual thresholds. The maximum of this global threshold and the absolute auditory threshold is then taken as the final threshold (Fig. 3.26, top).

MATLAB function involved:

- `function [SMR, min_threshold_subband, masking_threshold] = MPEG1_psycho_acoustic_model1(frame)`

computes the masking threshold (in dB) corresponding to psychoacoustic model #1 used in MPEG-1 audio (cf. ISO/CEI norm 11172-3:1993 (F), pp. 122–128). Input `frame` length should be 512 samples.
`min_threshold_subband` returns the minimum of `masking threshold` in each of the 32 sub-bands. `SMR` returns 27 signal-to-mask ratios (in dB); `SMR`(28–32) are not used.

```
frame=input_signal(11001:11512);
[SMR, min_threshold,frame_psd_dBSPL]= ...
    MPEG1_psycho_acoustic_model1(frame);
% NB: the power levels returned by this function assume that a
full-scale signal (in [-1,+1]) corresponds to 96 DB SPL

f = (0:255)/512*44100;
auditory_threshold_dB = 3.64*((f/1000).^-0.8) - ...
    6.5*exp(-0.6.*((f/1000)-3.3).^2) + 0.001*((f/1000).^4);
plot(f, frame_psd_dBSPL, f, min_threshold, 'r', ...
     f, auditory_threshold_dB, 'k');
hold off;
axis([0 22050 -20 100]);
legend('Signal PSD', 'Min. threshold per sub-band',...
       'Absolute threshold');
xlabel('Frequency (Hz)'); ylabel('Magnitude (dB)');
```

Signal-to-mask ratios (SMR) are computed for each band in a very conservative way, as the ratio of the maximum of the signal PSD to the minimum of the masking threshold in each band. Bit allocation is performed by an iterative algorithm, which gives priority to sub-bands with higher SMR. The resulting SNR in each sub-band should be greater than or equal to the SMR so as to push the noise level below the masking threshold (Fig. 3.26, bottom).

MATLAB function involved:

- `function [N_bits,SNR]= MPEG1_bit_allocation(SMR,bit_rate)` implements a simplified bit allocation greedy algorithm. `SMR` is the signal-to-mask ratios in each sub-band, as defined by the MPEG1 psycho-acoustic model. `bit_rate` is in kbps. `N_bits` is the number of bits in each sub-band. `SNR` is the maximum SNR in each sub-band after quantization, i.e., the SNR assuming each sub-band contains a full-range sinusoid. `N_bits` and `SNR` are set to zero for sub-bands 28–32.

```
% Allocating bits for a target bit rate of 192 kbits/s
% (compression ratio = 4)
[N_bits, SNR] = MPEG1_bit_allocation(SMR, 192000);

stairs((0:32)/32*22050,[SMR SMR(32)]);
hold on;
stairs((0:32)/32*22050,[SNR SNR(32)],'--');
axis([0 22050 -20 100]);
legend('SMR', 'SNR');
xlabel('Frequency (Hz)'); ylabel('Magnitude (dB)');
```

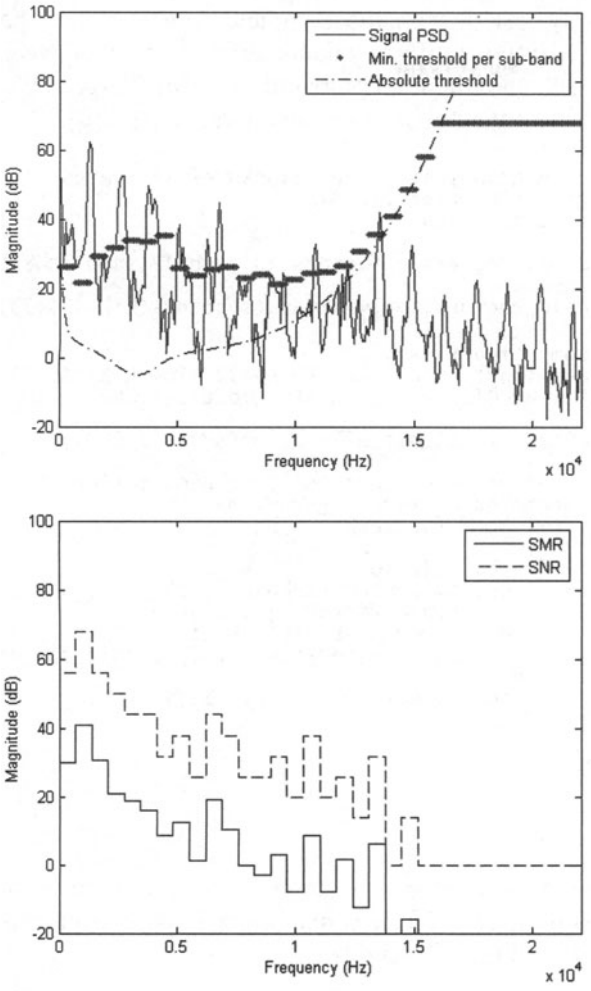

Fig. 3.26 The MPEG-1 psychoacoustic model #1. *Top*: Masking threshold per sub-band; *bottom*: signal-to-mask ratios (SMR) and the maximum (theoretical) SNR obtained after bit allocation. SNR is systematically above SMR so as to make quantization noise inaudible[12].

[12] Note that this is made possible only by the fact that we have allocated enough bits for quantizing frames. When the allowed bit rate decreases, SNR can be lower than SMR.

Let us finally test this on the complete signal, adding perceptual bit allocation to adaptive uniform quantization. Note that we simplify the quantization scheme here, compared to MPEG-1, by considering quantization with any number of bits in the range [0...16].

```
% Adaptive quantization per blocks of 12 frames
n_frames=fix(n_frames/12)*12;
for k=1:12:n_frames

    % Computing scale factors in each 12 samples sub-band
    % chunk
    [scale_factors,tmp]=max(abs(subbands(k:k+11,:)));

    % Computing SMRs [13]
    frame=input_signal(176+(k-1)*32:176+(k-1)*32+511);
    SMR = MPEG1_psycho_acoustic_model1(frame);

    N_bits = MPEG1_bit_allocation(SMR, 192000);

    % Adaptive perceptual uniform quantization, using a mid-
    % thread quantizer in [-Max,+Max]
    for j=1:32 % for each sub-band

        if N_bits(j)~=0
            codes= uencode(subbands(k:k+11,j),N_bits(j),...
                scale_factors(j),'signed');
            quantized_subbands(k:k+11,j) = ...
                udecode(codes,N_bits(j),scale_factors(j));
        else
            quantized_subbands(k:k+11,j) = 0;
        end;

    end;

end;
```

Quantization noise is now very small in some prominent sub-bands like sub-band #2 (Fig. 3.27, left). It is also more important in other sub-bands like sub-band #20 (Fig. 3.27, right).

[13] The input frame for the psychoacoustic model is delayed by 176 samples, as it should be centered in the middle of the local block of 12 sub-band samples, and the first sub-band sample corresponds to original sample 256 (actually to sample 512, but the analysis filter introduces a delay of 256 samples). So the center of the first frame should be on the original sample 256+11*32/2=432, and the beginning of this frame falls on sample 11*32/2=176.

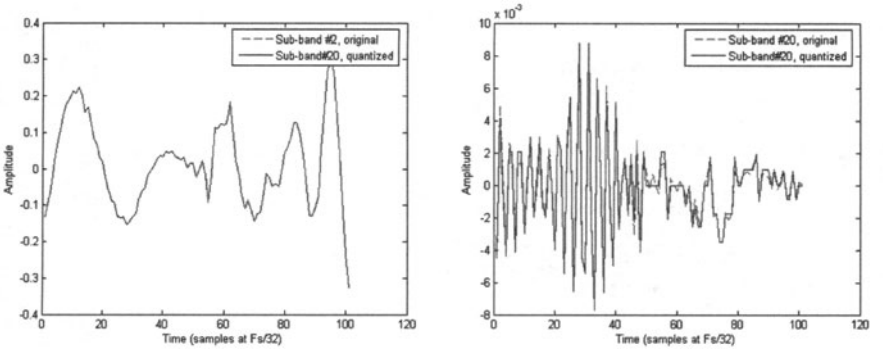

Fig. 3.27 Excerpts from sub-bands #2 and #20

The overall SNR has increased to 36.6 dB. The perceptual SNR is actually much higher, since most of the noise cannot be heard. The resulting signal is high-quality audio (Fig. 3.28).

```
snr_scaled_perceptual=snr(input_signal(512:end-512),...
    output_signal(512:end-512),0)
```

```
snr_scaled_perceptual =    36.6457
```

The price to pay is that the scale factors and the number of bits per sub-band must now be stored every 12 frames (i.e., every 12*32 sub-band samples). In the MPEG-1 norm, scale factors are expressed in decibels and quantized on 6 bits each. This comes from the fact that the ear perceives loudness as the log of the energy and has about 96 dB of hearing dynamics with a sensitivity threshold of about 1 dB. With 6 bits, the quantization step is 96/64 dB and the error lies in [−96/64/2, +96/64/2], which is below the 1 dB threshold. Assuming 4 bits are required for encoding the number of bits used in each of the 27 first sub-bands (sub-bands 28–32 are ignored in MPEG-1), this leads to 27*10=270 bits every 12 frames.[14] It is easy to obtain the number of bits used for the last block of 12 frames in our audio test and the related bit rate.

```
bits_per_block=sum(N_bits)*12+270
bit_rate=bits_per_block*44100/384
```

```
bits_per_block =    1674
bit_rate =  1.9225e+005
```

[14] In practice, MPEG-1 does not even allow all integer values in [1,16] for the number of bits in each band, which makes it possible to quantize it to less than 4 bits. The total number of bits used for bit allocation can then be reduced to 88 instead of 4*27=108.

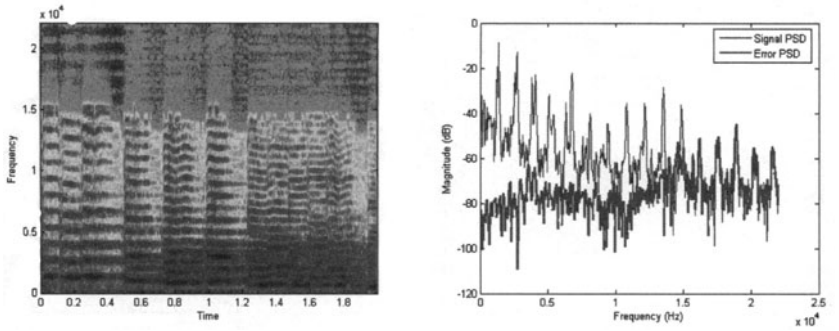

Fig. 3.28 Sub-band coding using adaptive perceptual uniform quantization. *Left*: spectrogram; *right*: periodogram of the original and error signals, around sample 11,000.

3.3 Going further

Unfortunately, MATLAB simulation code for audio coders is not widespread, with the remarkable exception of the toolbox designed by F. Petitcolas at Cambridge University (Petitcolas 2003).

Several other implementations (not in MATLAB) are available from the MPEG homepage (Chariglione 2007) and the MPEG.ORG website (MPEG TV 1998).

3.4 Conclusion

Although a bit tricky to implement correctly, perceptual audio coders are based on simple principles: perfect reconstruction filter banks and sub-band quantization driven by the masking effect.

If audio compression is one of the most prominent applications of sub-band (or transform) coding, it is by far not the only one. As early as in the 1950s, the idea of using the frequency-dependent sensitivity of the human eye to color (the eye is mostly sensitive to luminance, much less to chrominance) led to the definition of the NTSC encoding system in which most of the blue signal is discarded, keeping most of the green and only some of the red. More recently, the JPEG and MPEG/video coding standards were also based on transform coding, using the discrete cosine transform (see Chapters 8 and 9).

References

Chariglione L (2007) The MPEG Home Page [online] Available: http://www.chiariglione.org/mpeg/ [17/04/07]

Crochiere RE, Rabiner LR (1983) Multirate Digital Signal Processing. Englewood Cliffs, NJ: Prentice Hall.

Esteban D, Galand C (1977) Application of quadrature mirror filters to split-band voice coding schemes. Proc. IEEE Int. Conf. Acoustics, Speech and Signal Processing. Hartford, Connecticut, pp 191–195

Johnston JD (1980) A filter family designed for use in quadrature mirror filter banks. Proc. IEEE Int. Conf. Acoustics, Speech and Signal Processing, pp. 291–294

Malvar H (1990) Lapped transforms for efficient transform/subband coding. IEEE Trans. Acoust., Speech, Signal Processing, 38: 969–978

MPEG TV (1998) MPEG pointers and resources [online] Available: http://www.mpeg.org [17/04/07]

Nussbaumer HJ, Vetterli M (1984) Pseudo quadrature mirror filters. Digital Signal Processing, pp 8–12

Painter T, Spanias A (2000) Perceptual coding of digital audio. Proc. IEEE 88-4: 451–512

Pan D (1995) A tutorial on MPEG audio compression. IEEE Multimedia Magazine 2–2:60–74.

Petitcolas N (2003) MPEG for Matlab [online] Available: http://www.petitcolas.net/fabien/software/mpeg/index.html [17/04/07]

Princen J, Bradley A (1986) Analysis/synthesis filter bank design based on time domain aliasing cancellation. IEEE Trans. Acoust. Speech, Signal Processing 34: 1153–1161

Vaidyanathan PP (1993) Multirate Systems and Filter Banks. Englewood Cliffs, NJ: Prentice-Hall

Chapter 4

How does a dictation machine recognize speech?

T. Dutoit (°), L. Couvreur (°), H. Bourlard (*)

(°) Faculté Polytechnique de Mons, Belgium
(*) Ecole Polytechnique Fédérale de Lausanne, Switzerland

This chapter is not about how to wreck a nice beach[1]

There is magic (or is it witchcraft?) in a speech recognizer that transcribes continuous radio speech into text with a word accuracy of even not more than 50%. The extreme difficulty of this task, though, is usually not perceived by the general public. This is because we are almost deaf to the infinite acoustic variations that accompany the production of vocal sounds, which arise not only from physiological constraints (coarticulation) but also from the acoustic environment (additive or convolutional noise, Lombard effect) or from the emotional state of the speaker (voice quality, speaking rate, hesitations, etc.)[2]. Our consciousness of speech is indeed not stimulated until after it has been processed by our brain to make it appear as a sequence of meaningful units: phonemes and words.

In this chapter, we will see how statistical pattern recognition and statistical sequence recognition techniques are currently used for trying to mimic this extraordinary faculty of our mind (section 4.1). We will follow, in Section 4.2, with a MATLAB-based proof of concept of word-based automatic speech recognition (ASR) based on hidden Markov models (HMM), using a bigram model for modeling (syntactic–semantic) language constraints.

---

[1] It is, indeed, about *how to recognize speech.*

[2] Not to mention interspeaker variability or regional dialects.

T. Dutoit, F. Marqués (eds.), *Applied Signal Processing*,
DOI 10.1007/978-0-387-74535-0_4,