Chapter 7

How could music contain hidden information?

*W*hy not *A*ttempt *T*o hid*E* the very p*R*esence
of a *M*ess*A*ge in an audio t*R*ac*K ?*

C. Baras ($^{+}$), N. Moreau (*), T. Dutoit (°)

($^{+}$) GIPSA-lab (Grenoble Image Parole Signal et Automatique), Grenoble, France
(*) Ecole Nationale Supérieure des Télécommunications, Paris, France
(°) Faculté Polytechnique de Mons, Belgium

Audio watermarking started in the nineties as a modern and very technical version of playing cat and mouse[1]. The music industry, dominated by the "big four" record groups, also known as the "Majors" (Sony BMG, EMI, Universal and Warner), quickly realized that the availability of digital media for music recordings and the possibility to transfer it fast and degradation-free (thanks to the availability of high data-rate networks and of efficient data compression standards) would not only offer many benefits in terms of market expansion, but also expose their business to a great danger: that of piracy of intellectual property rights. Being able to insert proprietary marks in the media without affecting audio quality (i.e., in a

[1] The real origins of watermarking can be found in the old art of steganography (literally, "hidden writing", in ancient Greek). While the goal of cryptography is to render a message unintelligible, steganography attempts to hide the very presence of a message by embedding it into another information (Petitcolas *et al.* 1999)

"transparent way") was then recognized as a first step towards solving this issue. Additionally ensuring the robustness of the proprietary mark to not only usual media modifications (such as cropping, filtering, gain modification, or compression) but also to more severe piracy attacks quickly became a hot research topic worldwide.

Things changed radically in 2007 (i.e. before audio watermarking could be given a serious commercial trial). As a matter of fact, the Digital Rights Management (DRM) employed by the Majors to restrict the use of digital media (which did not make use of watermarking techniques) became less of a central issue after the last Majors stopped selling DRM-protected CDs in January 2007, and after S. Jobs' open letter calling the music industry to simply get rid of DRM, in February that same year (Jobs 2007).

Audio watermarking techniques are now targeted to more specific applications, related to the notions of "augmented content" or "hidden channel". One such application is audio fingerprinting[2], in which a content-specific watermark is inserted in the media, and can thus be retrieved for the automatic generation of the playlist of radio stations and music television channels for royalty tracking. Audio watermarking has also been proposed for the next generation of digital TV audience analysis systems: each TV channel would insert its own watermark in the digital audio and video streams it delivers, and a number of test TV viewers would receive a watermark detection system connected to a central computer[3].

In the next Section, we will first introduce the principles of spread spectrum and audio watermarking (7.1). This will be followed by a MATLAB-based proof of concept, showing how to insert a text message in an audio signal (7.2), and by a list of pointers to more advanced literature and software (7.3).

[2] The term *fingerprinting* is also used in the literature for specifying the technique which associates a short numeric sequence to an audio signal (i.e. some sort of *hashing;* Craver *et al*. 2001). We do not use it here with this meaning.

[3] With analog TV, audience analysis is much simpler: since each TV channel uses a specific carrier frequency, measuring this frequency through a sensor inserted in TV boxes gives the channel being watched at any time. This is no longer possible with digital TV, since several digital TV channels often share the same carrier frequency thanks to digital modulation techniques.

7.1 Background – Audio watermarking, seen as a digital communication problem

Watermarking an audio signal can be seen as a means of transmitting a sequence of bits ("1"s and "0"s) through a very particular noisy communication channel, whose "noise" contribution is the audio signal itself. With this view in mind, creating an audio watermarking system amounts to designing an emitter and a receiver adapted to the specificities of the channel: the emitter has to insert the watermark in the audio signal in such a way that the watermark cannot be heard, and the receiver must extract this hidden information from the watermarked audio signal[4] (Fig. 7.1).

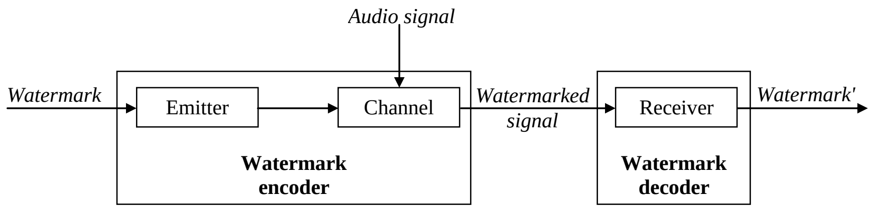

Fig. 7.1 Audio watermarking, seen as a communication problem

The problem therefore looks very similar to that of sending information bits through the internet with an ADSL connection: one has to maximize the bit rate while minimizing the bit error rate, even in the presence of communication noise and distortion[5]. In other words, watermarking directly benefits from fifty years of digital communication theory, after the pioneering work of Claude Shannon.

However, watermarking techniques are very specific in terms of bit rate, error rate, and signal-to-noise ratio. SNR is typically made very low, so as to make the watermark inaudible. This results in very low bit rates (several hundred bits per second at best) and high error rates (typically 10^{-3}), as opposed to the Megabits per second of ADSL modems and the associated low error rates (10^{-6}). In that respect, watermarking techniques are closer to those used for communication between interplanetary satellites and earth. In particular, spread spectrum techniques are used in both cases.

[4] This is often referred to as *blind* watermarking, in which the receiver has no *a priori* information on the audio signal.

[5] Watermark distortion is typically produced by filtering and encoding operations applied to the audio signal while being transmitted, such as MP3 encoding.

Spread spectrum signals will be reviewed in Section 7.1.1. In the next Sections, we will consider the emitter/receiver design issues in a blind (7.1.2), and informed watermarking system (7.1.3), respectively.

7.1.1 Spread spectrum signals

Spread spectrum communications (Peterson *et.al.* 1995) were originally used for military applications[6], given their robustness to narrow-band interference, their secrecy, and their security. As a matter of fact, spread spectrum signals are characterized by two unique features:

1. Their bandwidth is made much wider than the information bandwidth (typically 10 to 100 times for commercial applications; 1000 to 10^6 times for military applications). The spread of energy over a wide band results in lower power spectral density (PSD), which makes spread spectrum signals less likely to interfere with narrowband communications. Conversely, narrow band communications cause little to no interference to spread spectrum systems because their receiver integrates over a very wide bandwidth to recover a spread spectrum signal. As we shall see below, another advantage of such a reduced PSD is that spread spectrum signal PSD can be pushed on purpose below communication noise, thereby making these signals unnoticeable.
2. Some *spreading sequence* (also called *spreading code* or *pseudo-noise*) is used to create the wide-band spread spectrum signal from the information bits to send. This code must be shared by the emitter and the receiver, which makes spread spectrum signals hard to intercept.

The price to pay for these features is a reduced spectral efficiency, defined as the ratio between the bit rate and the bandwidth: the number of bits per second and per Hertz provided by spread spectrum techniques is typically less than 1/5, while "standard" techniques offer a spectral efficiency close to 1 (Massey 1994).

In the nineties, spread spectrum techniques have found a major application in Code Division Multiple Access (CDMA), now used in satellite positioning systems (GPS, hopefully in Galileo), and for wireless digital phone communications systems (UMTS).

Direct sequence spread spectrum (DSSS)

Common spread spectrum systems are of the *time hopping* (also called *direct sequence*) or *frequency hopping* type. In the latter case, as its name

[6] Spread spectrum communications dates back to World War II.

implies, the carrier of a frequency hopping system hops from frequency to frequency over a wide band, in an order defined by the spreading code. In contrast, direct sequence spread spectrum (DSSS) signals are obtained by multiplying each bit-sized period (of duration T_b) of the emitted signal (the watermark signal in our case) by a (unique) spreading sequence, composed of a random sequence of ±1 rectangular pulses (of duration T_c) (Fig. 7.2, left).

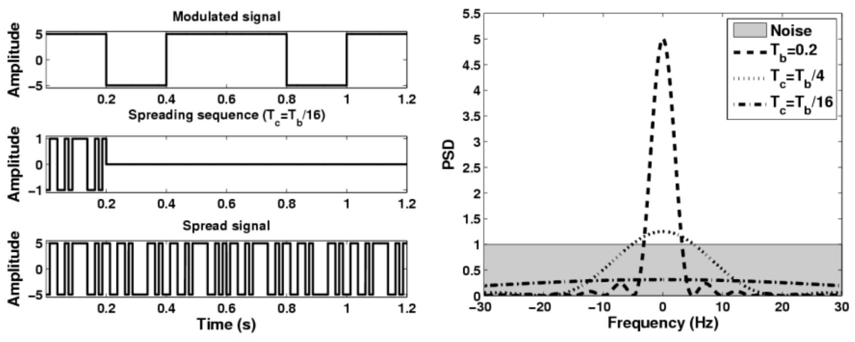

Fig. 7.2 Left: Emitted signal, spreading sequence, and spread signal waveforms, for T_b=0.2s and T_c=T_b/16. Right: PSD of the input signal, compared to that of the spread signal for T_c=T_b/4 and T_c=T_b/16.

It is easy to compute the PSD of such a spread spectrum signal if we assume that the emitted signal is a classical NRZ random binary code composed of rectangular pulses of duration T_b and of amplitude $\pm 1/T_b$. As a matter of fact, the PSD of such a signal is given by:

$$S_{NRZ}(f) = \frac{1}{T_b}\operatorname{sinc}^2(fT_b) \tag{7.1}$$

and since the spread signal is itself an NRZ signal with pulse duration T_c and pulse amplitude $\pm 1/T_b$, its PSD is given by:

$$S_{spread}(f) = \frac{T_c}{T_b^{\,2}}\operatorname{sinc}^2(fT_c) \tag{7.2}$$

Both PSDs are plotted in Fig. 7.2 (right), for two values of T_c. Clearly, when T_c decreases, the bandwidth of the spread signal increases, and its maximum PSD level decreases accordingly.

As announced above, the resulting spread spectrum signal offers an increased robustness to narrow-band interference and distortion: while a low-pass filter with cut-off frequency of 5 Hz would almost completely delete the input NRZ signal, its effect on the spread signal would be mild. Spread spectrum techniques also make it possible to hide information in communication noise, as suggested in Fig. 7.2 (right) with a background noise of unitary variance.

7.1.2 Communication channel design

In this Section, we will discuss emitter and receiver design issues, and estimate the corresponding bit error rate.

Emitter

We will use the notations introduced in Fig. 7.3: the audio signal *x(n),* sampled at F_s Hz, is added to a spread spectrum watermark signal *v(n),* obtained by modulating a watermark sequence of *M* bits b_m (in {0,1}), with a spreading signal *c(n)* composed of N_b samples (and shown in Fig. 7.3 as a vector $\mathbf{c}=[c(0),c(1),\ldots,c(N_b\text{-}1)]^T$).

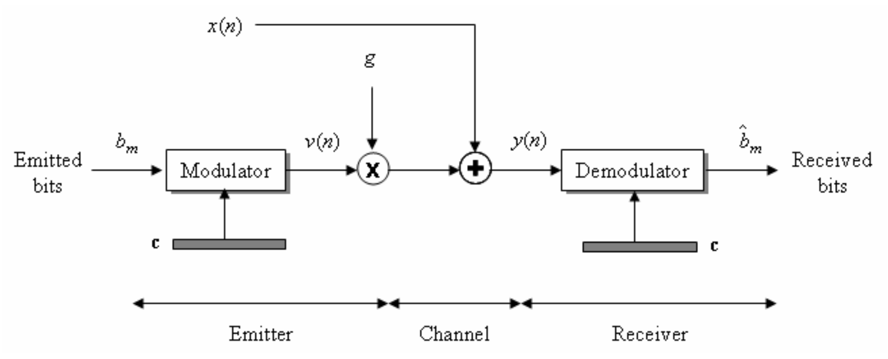

Fig. 7.3 A more detailed view of a watermarking system, seen as a communication system with additive channel noise

Vector **c** is synthesized by a pseudo-random sequence generator, as a Walsh-Hadamard sequence or a Gold sequence with values in {-1,+1}.

Following the time-hopping principle exposed in the previous Section, the spread spectrum signal *v(n)* is given by:

$$v(n) = \sum_{m=0}^{M-1} a_m c(n - mN_b) \tag{7.3}$$

where a_m is a symbol in {-1,+1} given by $a_m=2b_m-1$. One can also see this signal as a concatenation of vectors:

$$\mathbf{v_m} = a_m\mathbf{c} = \begin{cases} +\mathbf{c} & if\ a_m = 1 \\ -\mathbf{c} & if\ a_m = -1 \end{cases} \tag{7.4}$$

It is finally amplified by some gain *g,* which makes it possible to control the signal-to-noise ratio (SNR) between the watermarked signal *v(n)* and the audio signal *x(n).*

Since one bit is emitted every N_b samples, the bit-rate is simply given by $R=F_s/N_b$ (bits/s), where F_s is the sampling frequency (typically 44.1 kHz).

Receiver

In case of the additive white Gaussian noise (AWGN) channel shown in Fig. 7.3, the signal *y*(*n*) available at the receiver is a noisy version of the emitted signal *v(n)*, with channel noise PDF given by $N(0,\sigma_X^2)$.

We will further assume that the received samples *y(n)* are perfectly synchronized with the emitted samples *x(n)*, i.e., that the received samples can be organized in vectors $\underline{y}_m=[y(mN_b), y(mN_b+1), \ldots, y(mN_b+N_b-1)]^T$ related to vectors $\underline{v}_m$ by:

$$\mathbf{y_m} = g\mathbf{v_m} + \mathbf{x_m} = \begin{cases} +g\mathbf{c} + \mathbf{x_m} & if\ a_m = 1 \\ -g\mathbf{c} + \mathbf{x_m} & if\ a_m = -1 \end{cases} \tag{7.5}$$

where $\mathbf{x_m}=[x(mN_b), x(mN_b+1), \ldots, x(mN_b+N_b-1)]^T$ is a vector of channel noise samples, and *m* is the frame index.

Since the modulation operation is assimilated to raw concatenation[7], and thereby produces no symbol interference, symbol detection can be performed on a symbol-by-symbol basis (no Viterbi algorithm is needed). The optimal detector in this case is the *Maximum a Posteriori* (MAP) detector (Proakis 2001)[8], which maximizes the probability of each received bit $\tilde{b}_m$ given the corresponding received vector $\mathbf{y_m}$:

[7] This is the simplest possible case of a memoryless linear modulation scheme.

[8] The same as the one used in Chapter 4 for vowel classification.

$$\tilde{b}_m = \underset{b\in\{0,1\}}{\arg\max}\, P(b\,|\,\mathbf{y_m}) \tag{7.6}$$[9]

In the next paragraphs, we will show that this is equivalent to estimating $\tilde{b}_m$ from the sign of the normalized scalar product α_m between the received vector $\mathbf{y_m}$ and the spreading code $\mathbf{c}$:

$$\alpha_m = \frac{<\mathbf{y_m},\mathbf{c}>}{\|\mathbf{c}\|^2} = \frac{<\mathbf{y_m},\mathbf{c}>}{N_b} \tag{7.7}$$

As a matter of fact, following the same development as in Chapter 4, Section 4.1, equation (7.6) is equivalent to:

$$\begin{aligned}\tilde{b}_m &= \underset{b\in\{0,1\}}{\arg\max}\, \frac{P(\mathbf{y_m}\,|\,b)P(b)}{P(\mathbf{y_m})} \\ &= \underset{b\in\{0,1\}}{\arg\max}\, P(\mathbf{y_m}\,|\,b)P(b)\end{aligned} \tag{7.8}$$

in which $P(\mathbf{y_m})$ is the *a priori* probability of $\mathbf{y_m}$, which does not influence the estimation of $\tilde{b}_m$, and $P(b)$ is the *a priori* probability of each possible value of b. Furthermore, if the symbols have the same probability of occurrence, (7.8) reduces to:

$$\tilde{b}_m = \underset{b\in\{0,1\}}{\arg\max}\, P(\mathbf{y_m}\,|\,b) \tag{7.9}$$

known as the *maximum likelihood* decision.

In the AWGN channel case, since $\mathbf{x_m}$ is an N_b-dimensional multivariate Gaussian random variable with zero mean and covariance matrix equal to σ_x^2I (where I is the N_b-dimensional unity matrix), equation (7.5) shows that $\mathbf{y_m}$ is also a multivariate Gaussian random variable, with mean equal to $a_m g\mathbf{c}$ and the same covariance matrix as $\mathbf{x_m}$. Hence:

$$P(\mathbf{y_m}\,|\,b) = \frac{1}{\sqrt{(2\pi\sigma_X^2)^{N_b}}} \exp\left(-\frac{\|\mathbf{y_m}-(2b-1)g\mathbf{c}\|^2}{2\sigma_X^2}\right) \tag{7.10}$$

[9] In the sequel, we use the same shortcut notation for probabilities as in Chapter 4: $P(a|b)$ denotes the probability $P(A=a|B=b)$ that a discrete random variable A is equal to a given the fact that random variable B is equal to b will simply be written, or the probability *density* $p_{A|B=b}$ (a)) when A is a continuous random variable.

It follows that:

$$\begin{aligned}\tilde{b}_m &= \underset{b\in\{0,1\}}{\arg\min} \|\mathbf{y_m} - (2b-1)g\mathbf{c}\|^2 \\ &= \underset{b\in\{0,1\}}{\arg\min} \left(\|\mathbf{y_m}\|^2 - 2 < \mathbf{y_m}, (2b-1)g\mathbf{c} > + (2b-1)^2 \|\mathbf{c}\|^2\right) \\ &= \underset{b\in\{0,1\}}{\arg\max} < \mathbf{y_m}, (2b-1)g\mathbf{c} >\end{aligned} \qquad (7.11)$$

since $(2b\text{-}1)^2 = 1$ whatever the value of b in $\{0,1\}$. In other words, as announced, $\tilde{b}_m$ is obtained from the sign of the scalar product $<\mathbf{y_m}, \mathbf{c}>$, i.e., from the sign of α_m:

$$\tilde{b}_m = \begin{cases} 0 & \text{if} < \mathbf{y_m}, \mathbf{c} > \leq 0 \\ 1 & \text{if} < \mathbf{y_m}, \mathbf{c} > > 0 \end{cases} \qquad (7.12)$$

Fig. 7.4 shows a synthetic view of (7.9) in case b_m is "1". In the left plot, $\tilde{b}_m$ = "1", while the right plot will wrongly result in $\tilde{b}_m$ = "0".

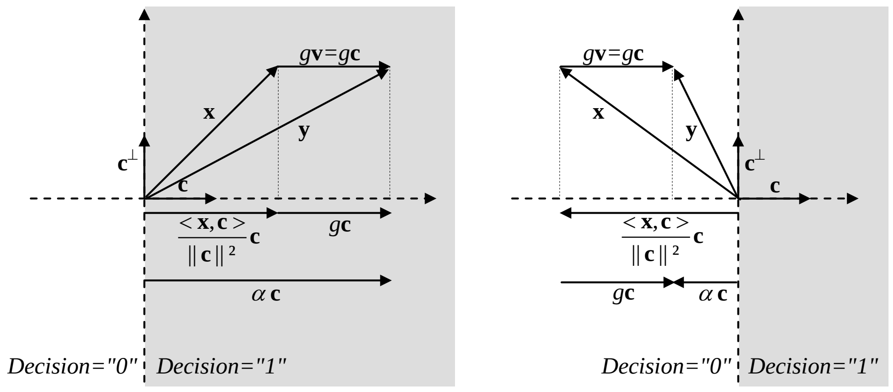

Fig. 7.4 Watermarking of audio vector **x** with symbol "1". Watermarked vector **y** is obtained by adding $g\mathbf{v}$ (i.e., $g\mathbf{c}$, given the symbol is "1") to **x**. Symbol "1" will be detected if $\alpha \geq 0$. Left: since $<\mathbf{x},\mathbf{c}>$ is positive, any (positive) value of g will result in $\alpha \geq 0$. Right: since $<\mathbf{x},\mathbf{c}>$ is negative, only a value of $g \geq <\mathbf{x},\mathbf{c}>/\|\mathbf{c}\|^2$ will result in $\alpha \geq 0$ (this is not the case on the figure).

Error rate

The PDF of α_m for g set to 1 and σ_x^2 set to 20 dB is given in Fig. 7.5 for various vector lengths N_b, i.e. for various bit rates R.

Projecting Equation (7.5) on the spread sequence **c**, the scalar product α_m appears as a one-dimensional Gaussian random variable with mean $(2b_m\text{-}1)g$ and variance $\sigma_x^2 / \|\mathbf{c}\|^2$. The latter is that of the normalized scalar product between the N_b-dimensional multivariate Gaussian random variable $\mathbf{x_m}$ and **c**. Hence:

$$P(\alpha_m \mid b_m) = \frac{1}{\sqrt{2\pi}\sigma_X / \|\mathbf{c}\|} \exp\left(-\frac{\alpha_m - (2b_m - 1)g}{2\sigma_X^2 / \|\mathbf{c}\|^2} \right) \tag{7.13}$$

As expected, the probability for α_m to be positive while b_m=0 and the related probability for α_m to be negative while b_m=1 are non-zero, and these values increase with R (i.e., when N_b decreases). The probability of error is given by:

$$P_e = P(b=1)P(\alpha \leq 0 \mid b=1) + P(b=0)P(\alpha > 0 \mid b=0) \tag{7.14}$$

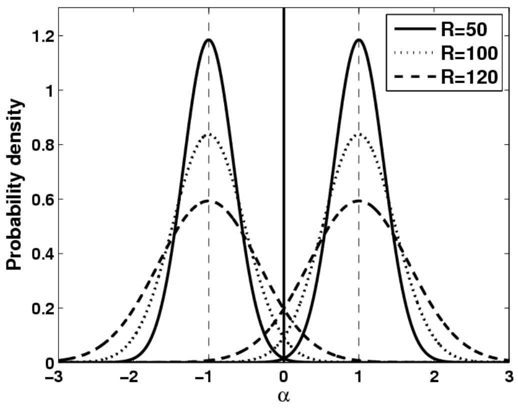

Fig. 7.5 Probability density functions for α when b=0 and when b=1, for various values of the bit rate R (and with F_s=44100, g=1, σ_x^2 =20dB)

When symbols have the same *a priori* probability, this probability of error can be estimated on Fig. 7.5 as the area under the intersection of $P(\alpha_m|b_m\text{=}0)$ and $P(\alpha_m|b_m\text{=}1)$. It is also possible to compute P_e analytically (Proakis 2001), which leads to:

$$P_e = Q\left(\sqrt{\frac{g^2 \| c \|^2}{\sigma_X^2}}\right) = Q\left(\sqrt{\frac{F_s}{R}\frac{g^2\sigma_C^2}{\sigma_X^2}}\right) \quad \textit{with} \quad Q(u) = \int_u^\infty \frac{1}{\sqrt{2\pi}} \exp\left(-\frac{t^2}{2}\right) dt \tag{7.15}$$

in which we have used $\sigma_C^2 = \|\mathbf{c}\|^2 / N_b$, an estimate of the power of the spreading sequence.

The probability of error is thus finally a function of the watermarking SNR (i.e., the ratio between the power of the emitted signal, $g^2\sigma_C^2$, and that of the audio signal, σ_X^2) and of the bit rate R. Fig. 7.6 shows a plot of $Q\left(\sqrt{u}\right)$ and the resulting plot of $P_e(R)$ for various values of the SNR. It appears that $P_e=10^{-3}$ can be reached with a bit rate of 50 bits/s and a SNR of -20 dB. As we will see below, this SNR approximately corresponds to a watermark just low enough not to be heard (or, consequently, to the kind of SNR introduced by MPEG compression; see Chapter 3).

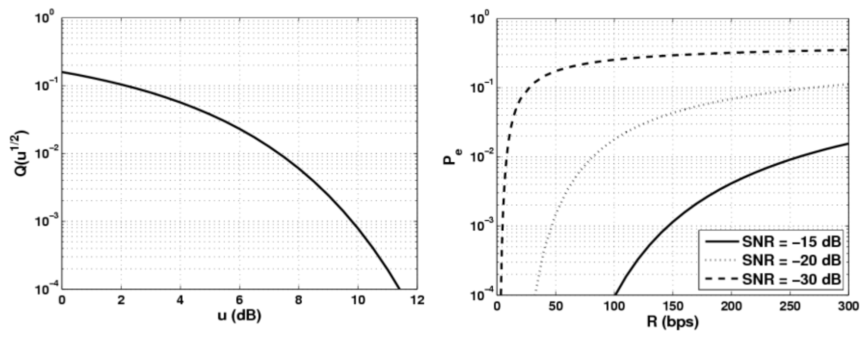

Fig. 7.6 Left: $Q\left(\sqrt{u}\right)$ with u in dB; Right: $P_e(R)$ for various values of the SNR and Fs=44100

7.1.3 Informed watermarking

While it is generally not possible to adapt the encoder to the features of the channel noise[10], such a strategy is perfectly possible in watermarking systems, as the communication noise itself is available at the encoder. In other

[10] In some communications systems, (like in ADSL modems), the communication SNR is estimated at the receiver, and sent back to the emitter through a communication backchannel.

words, watermarking is a good example of a communication channel with side information (Shannon 1958, Costa 1983). This opens the doors to the second generation of watermarking systems: that of *informed watermarking* (Cox *et al.* 1999, Chen and Wornwell 2001, Malvar and Florencio 2003) and its triple objectives: maximal bit-rate, minimal distorsion, maximal robustness. In the next paragraphs we will consider two steps in this direction: gain adaptation, and perceptual filtering.

Gain adjustment for error-free detection

The most obvious way to turn the encoder into an informed encoder is to adapt the gain g as a function of time so as to compensate for the sudden power variations in the audio signal and keep the SNR constant. This can be achieved by estimating σ_X^2 on audio frames (typically 20 ms long, in which music is assumed to be stationary) and imposing g to be proportional to σ_X.

It is also possible, and more interesting, to adjust g to impose *error-free* detection (Malvar and Florencio 2003, LoboGuerrero *et al.* 2003). As we have seen in the previous Section indeed, the probability of error can be made as small as we want, by shifting apart the Gaussians in Fig. 7.5, i.e., by increasing g. It is even possible to obtain error-free detection by adapting the gain to each frame (hence we will denote it as g_m) so as to yield:

$$\alpha_m = \frac{<\mathbf{y_m},\mathbf{c}>}{\|\mathbf{c}\|^2} = a_m g_m + \underbrace{\frac{<\mathbf{x_m},\mathbf{c}>}{\|\mathbf{c}\|^2}}_{\beta_m} = \begin{cases} \geq 0 & if\ a_m = 1 \\ < 0 & if\ a_m = -1 \end{cases} \tag{7.16}$$

As seen in Fig. 7.4, this condition is automatically verified for any (positive) value of g_m, if β_m has the same sign as a_m. In the opposite case, it is verified by imposing $g_m = -\beta_m / a_m$.

In practice though, the transmission channel itself will add noise to the watermarked vector $\mathbf{y_m}$, leading to received vectors equal to $\mathbf{y_m}+\mathbf{p_m}$. Condition (7.16) becomes:

$$\begin{aligned} \alpha_m &= <\mathbf{y_m}+\mathbf{p_m},\mathbf{c}> \\ &= a_m g_m + \underbrace{\frac{<\mathbf{x_m},\mathbf{c}>}{\|\mathbf{c}\|^2}}_{\beta_m} + \underbrace{\frac{<\mathbf{p_m},\mathbf{c}>}{\|\mathbf{c}\|^2}}_{\gamma_m} = \begin{cases} \geq 0 & if\ a_m = 1 \\ < 0 & if\ a_m = -1 \end{cases} \end{aligned} \tag{7.17}$$

Some security margin Δ_g is then required to counterbalance the projection of $\mathbf{p_m}$ on $\mathbf{c}$ (Fig. 7.7):

$$\begin{cases} g_m + \beta_m \geq \Delta_g & if\ a_m = 1 \\ -g_m + \beta_m \leq -\Delta_g & if\ a_m = -1 \end{cases} \tag{7.18}$$

The value of Δ_g is set according to the estimated variance of the transmission channel noise.

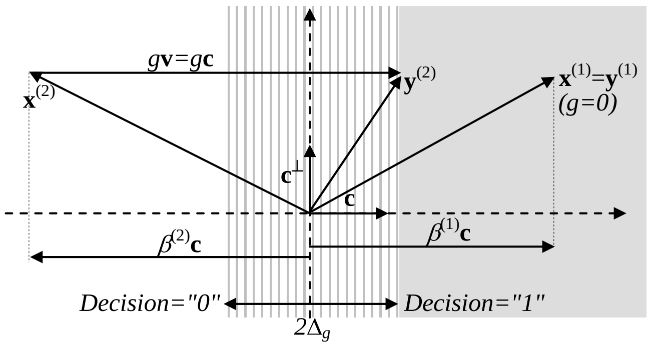

Fig. 7.7 Informed watermarking of audio vector **x** with symbol "1". Watermarked vector **y** is obtained by adding $g\mathbf{v}$ (i.e., $g\mathbf{c}$, given the symbol is "1") to **x**, with g chosen such that **y** falls in the gray region, where decision will be "1". Two cases are shown: for $\mathbf{x}^{(1)}$, g can be set to zero; for $\mathbf{x}^{(2)}$, g must have a high value. The gray region has been shifted from the white region by a $2\Delta_g$-wide gap, to account for possible transmission channel noise.

The efficiency of this watermarking strategy stems from the fact that the gain g is directly proportional to the correlation between the audio signal **x** and the spread sequence **c**. However, it does not ensure the perceptual transparency of the watermark: in case **x** is very much in the opposite direction of **c** while the emitted symbol is "1" (as it is the case of $\mathbf{x}^2$ in Fig. 7.7), g must be set to a high value. Chances are in such a case that the watermark may significantly distort the audio signal.

Perceptual shaping filter

A solution to the perceptual transparency requirement is to use properties of the human auditory system to ensure inaudible watermarking. As we have shown in Chapter 3, psychoacoustic modeling (PAM) has established the existence of a masking threshold, assimilated to a PSD $\Phi(f)$ (see Chapter 3, Sections 3.1.3 and 3.1.4) below which the PSD of the watermark $S_W(f)$ must lie for making it inaudible. This threshold is signal-dependent and must therefore be updated frequently, typically every 20 ms.

Since the efficiency of a watermarking system is a function of the SNR ratio (Fig. 7.6), the best performance is reached when:

$$S_W(f) = \Phi(f) \tag{7.19}$$

This can be achieved, as shown in Fig. 7.8, by replacing the scalar gain *g* by a perceptual shaping filter *G*(*f*), which is conveniently constrained to be an all-pole filter :

$$G(z) = \frac{b_0}{1 + a_1 z^{-1} + a_2 z^{-2} + \ldots + a_p z^{-p}} \tag{7.20}$$

If the input *v*(*n*) of this filter is assimilated to white noise (with zero mean and unity variance), adjusting the coefficients of this filter so that its output *w*(*n*) will have a given PSD $S_W(f)$ is a typical filter synthesis problem, which has already been examined in Chapter 1. It is based on solving a set of *p* linear equations, the so-called *Yule-Walker* equations (1.5), whose coefficients are the first *p* values of the autocorrelation function $\phi_w(k)$ of *w*(*n*). These values are related to the PSD of *w*(*n*) by the inverse discrete-time Fourier transform formula:

$$\phi_w(k) = \int_{-1/2}^{1/2} \Phi(f) e^{j2\pi fk} df \tag{7.21}$$

which can be approximated by an inverse discrete Fourier transform using *N* samples of Φ(*f*):

$$\phi_w(k) \simeq \frac{1}{N} \sum_{n=0}^{N-1} \Phi\left(\frac{n}{N}\right) e^{j\frac{2\pi}{N}nk} \tag{7.22}$$

The receiver has to be modified accordingly (Fig. 7.8), by inserting an inverse, zero-forcing equalizer $1/G(f)$ before the demodulator (Larbi *et al,* 2004). However, since the audio signal *x*(*n*) used to derive *G*(*f*) is not available at the receiver, an estimate $1/\tilde{G}(f)$ is computed by psychoacoustic modeling of the watermarked signal *y*(*n*). The accuracy of this estimation depends on the robustness of the PAM to the watermark *and* to possible transmission channel noise, such as MPEG compression. This specific constraint on the PAM used in watermarking distinguishes it from the PAMs used in audio compression (such as in MPEG).

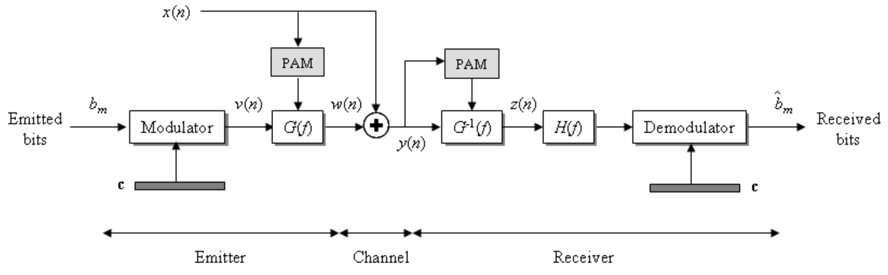

Fig. 7.8 Informed watermarking using a psychoacoustic model (PAM), a perceptual shaping filter *G*(*f*), and a Wiener equalizer *H*(*f*).

Once perceptual shaping has been inverted, detection is achieved as previously, except it is based on estimating the sign of <$\mathbf{z_m}$,**c**>, where *z(n)* is the output of the inverse filter. Assuming $1/\tilde{G}(f)$ is close enough to $1/G(f)$, $\mathbf{z_m}$ is given by:

$$\mathbf{z_m} = \mathbf{v_m} + \mathbf{r_m} \tag{7.23}$$

where *r*(*n*) is the audio signal *x*(*n*) filtered by $1/\tilde{G}(f)$. Equation (7.23) simply replaces equation (7.5).

Wiener filtering

As we will see in Section 7.2.3, the SNR at the output of the zero-forcing equalizer is very low: the power of the spread spectrum signal *v(n)* is small compared to that of the equalized audio signal *r(n).* Since the PSDs of *v(n)* and *r(n)* are different, it is possible to filter *z(n)* so as to increase the SNR, by enhancing the spectral components of *z(n)* dominated by *v(n)* and attenuating those dominated by *r(n).* This can even be done optimally by a symmetric FIR *Wiener filter H(z)* at the output of the zero-forcing equalizer (Fig. 7.8):

$$H(z) = \sum_{i=-p}^{p} h(i) z^{-i} \tag{7.24}$$

The output of the FIR Wiener filter *H(z)* is made maximally similar to *v(n),* in the minimum mean square error (MMSE) sense (Fig. 7.9): its coefficients are computed so as to minimize the power of the error signal *e(n)=z(n)-v(n)*. It can be shown (Haykin, 1996) that this condition is met if the coefficients *h(i)* of the filter are the solution of the so-called *Wiener-Hopf equations*:

$$\begin{bmatrix} \phi_{zz}(0) & \phi_{zz}(1) & \dots & \phi_{zz}(2p) \\ \phi_{zz}(-1) & \phi_{zz}(0) & \dots & \phi_{zz}(2p-1) \\ \dots & \dots & \dots & \dots \\ \phi_{zz}(-2p) & \phi_{zz}(-2p+1) & \dots & \phi_{zz}(0) \end{bmatrix} \begin{bmatrix} h_{-p} \\ h_{-p+1} \\ \dots \\ h_p \end{bmatrix} = \begin{bmatrix} \phi_{vv}(-p) \\ \phi_{vv}(-p+1) \\ \dots \\ \phi_{vv}(p) \end{bmatrix} \quad (7.25)$$

where $\phi_{zz}(k)$ and $\phi_{vv}(k)$ are the autocorrelation functions of *z(n)* and *v(n),* respectively.

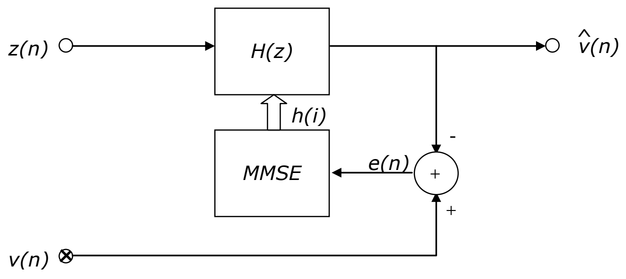

Fig. 7.9 Conceptual view of a Wiener filter [11]

In practice, since *r(n)* is not stationary, the Wiener filter is regularly adapted.

7.2 MATLAB proof of concept: ASP_watermarking.m

In this Section, we develop watermarking systems whose design is based on classical communication systems using spread spectrum modulation. We first examine the implementation of a watermarking system and give a brief overview of its performance in the simple and theoretical configuration where the audio signal is a white Gaussian noise (7.2.1). Then, we show how this system can be adapted to account for the audio signal specificities, while still satisfying the major properties and requirements of wa-

[11] In practice, *v(n)* is not available. Nevertheless, only its autocorrelation is required to solve the Wiener-Hopf equations.

termarking applications (namely inaudibility, correct detection and robustness of the watermark): we extend the initial system by focusing on the correct detection of the watermark (7.2.2), on the inaudibility constraint (7.2.3), and on the robustness of the system to MP3 compression (7.2.4).

7.2.1 Audio watermarking, seen as a digital communication problem

The following digital communication system is the direct implementation of the theoretical results detailed in Section 7.1. The watermark message is embedded in the audio signal at the binary rate `R`=100 bps. To establish the analogy between watermarking system and communication channel, the audio signal is viewed as the channel noise. In this section, it will therefore be modeled as a white Gaussian noise with zero mean and variance `sigma2_x`. `sigma2_x` is computed so that the Signal-to-Noise Ratio (SNR), i.e. the ratio between the watermark and the audio signal powers, is equal to -20 dB.

Emitter/Embedder

Let us first generate a watermark message, and its corresponding bit sequence.

```
message = 'Audio watermarking: this message will be embedded
in an audio signal, using spread spectrum modulation, and
later retrieved from the modulated signal.'

bits = dec2bin(message); % converting ascii to 7-bit string
bits = bits(:)'; % reshaping to a single string of 1's and 0's
bits = double(bits) - 48; % converting to a 0's and 1's
symbols = bits*2-1;  % a (1x1050) array of -1's and +1's
N_bits = length(bits);
```

```
message =

Audio watermarking: this message will be embedded in an audio
signal, using spread spectrum modulation, and later retrieved from
the modulated signal.
```

We then generate the audio signal, the spread spectrum sequence, and the watermarked signal. The latter is obtained by first deriving a symbol sequence (-1's and +1's) from the input bit sequence (0's and +1's), and then by modulating the spread spectrum signal by the symbol sequence (Fig. 7.10, left). This is achieved by concatenating spread spectrum waveforms weighted by the symbol values.

```
% Random sequences generators initialization
rand('seed', 12345);
randn('seed', 12345);

% Sampling frequency |Fs|, binary rate |R|, samples per bit
% |N_samples| and |SNR|
Fs = 44100;
R = 100;
N_samples = round(Fs/R);
SNR_dB = -20;

% Spread waveform: a random sequence of -1's and +1's
spread_waveform = 2*round( rand(N_samples, 1) ) - 1;

% Audio signal: a Gaussian white noise with fixed variance
sigma2_x = 10^(-SNR_dB/10);
audio_signal = sqrt(sigma2_x)*randn(N_bits*N_samples, 1);

% Modulated signal and watermark signal
modulated_signal = zeros(N_bits*N_samples, 1);
for m = 0:N_bits-1
    modulated_signal(m*N_samples+1:m*N_samples+N_samples)= ...
       symbols(m+1)*spread_waveform;
end
watermark_signal = modulated_signal; % no gain applied

% Plotting the baseband signal (i.e., the emitted signal
% corresponding to the input bit sequence) and the watermark
% signal

emitted_signal = ones(N_samples, 1)*symbols;
emitted_signal = emitted_signal(:);

axe_ech = (4*N_samples:7*N_samples+1);
subplot(2,1,1);
plot(axe_ech/Fs, emitted_signal(axe_ech));
subplot(2,1,2);
plot(axe_ech/Fs, watermark_signal(axe_ech));
```

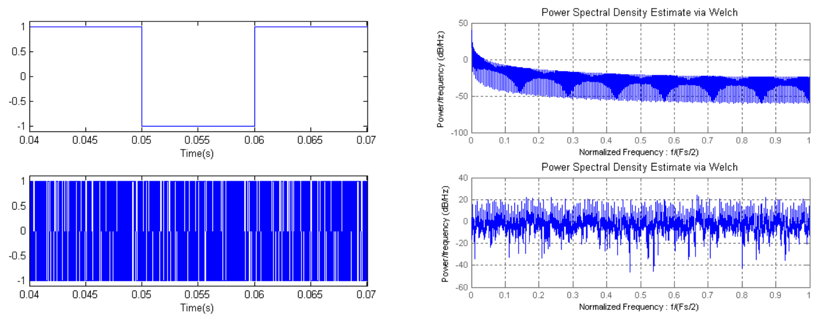

Fig. 7.10 Left: three symbols in {-1,+1} and the resulting spread spectrum waveforms, or watermark signal; Right : the corresponding PSDs

As expected, the power spectral density (PSD) of the spread spectrum sequence (i.e., the watermark signal) is much flatter than that of the baseband signal (i.e., the emitted signal) corresponding to the input bit sequence (Fig. 7.10, right), while their powers are identical: 0dB.

```
power_baseband_dB=10*log10(var(emitted_signal))
power_spread_spectrum_dB=10*log10(var(watermark_signal))

subplot(2,1,1)
pwelch(emitted_signal,[],[],[],2); 12
subplot(2,1,2)
pwelch(watermark_signal,[],[],[],2);
```

```
power_baseband_dB =  -0.0152
power_spread_spectrum_dB =  5.5639e-006
```

The embedding process finally consists in adding the result to the audio signal, yielding the audio watermarked signal. Plotting the first 100 samples of the watermark, audio, and watermarked signals shows that the watermarked signal is only slightly different from the audio signal, given the SNR we have imposed (Fig. 7.11).

```
watermarked_signal = watermark_signal + audio_signal;

subplot(3,1,1); plot((1:100)/Fs,watermark_signal(1:100),'k-');
subplot(3,1,2);plot((1:100)/Fs,audio_signal(1:100));
subplot(3,1,3);plot((1:100)/Fs,watermarked_signal(1:100),'r');
```

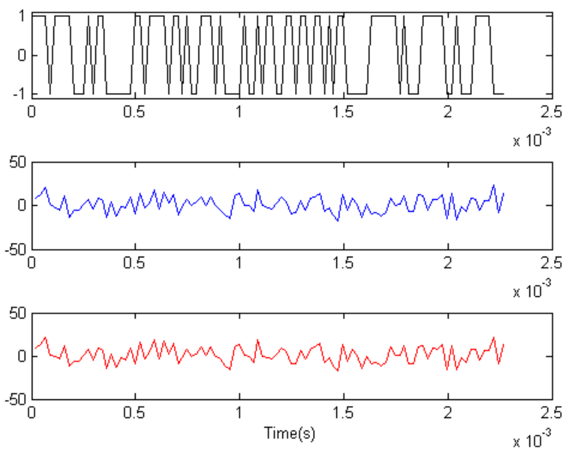

Fig. 7.11 Zoom on 100 samples. Top: watermark signal; Center: audio signal; Bottom: watermarked signal

[12] Notice that we claim F_s=2, so as to make `pwelch` return |FFT²/N|, in which the variance of a white noise is directly readable. See the appendix in `ASP_audio_cd.m` for more details.

Receiver

The watermark receiver is a correlation demodulator. It computes the normalized scalar product `alpha` between frames of the received signal and the spread spectrum waveform, and decides on the received bits, based on the sign of `alpha`. One can see on the plot (for bits 81 to 130) that the watermark sometimes imposes a wrong sign to `alpha`, leading to erroneous bit detection (Fig. 7.12).

```
alpha = zeros(N_bits, 1);
received_bits = zeros(1,N_bits);
for m = 0:N_bits-1
    alpha(m+1) =
(watermarked_signal(m*N_samples+1:m*N_samples+N_samples)' ...
        * spread_waveform)/N_samples;
    if alpha(m+1) <= 0
        received_bits(m+1) = 0;
    else
        received_bits(m+1) = 1;
    end
end

% Plotting the input bits, alpha, and the received bits.
range=(81:130);
subplot(3,1,1);
stairs(range, bits(range));
subplot(3,1,2);
stairs(range, alpha(range));
subplot(3,1,3);
stairs(range, received_bits(range));
```

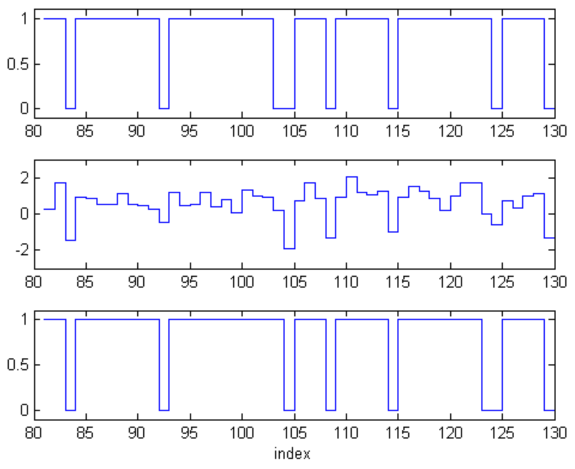

Fig. 7.12 Zoom on bits 80 to 130. Top: input bits; Center: `alpha`; Bottom: received bits

The performance of the receiver can be estimated by computing the Bit Error Rate (BER) and by decoding the received message. As can be observed on the message, a bit error rate of 0.02 is disastrous in terms of message understandability.

```
number_of_erroneous_bits= sum(bits ~= received_bits)
total_number_of_bits=N_bits
BER = number_of_erroneous_bits/N_bits
received_chars = reshape(received_bits(1:N_bits),N_bits/7, 7);
received_message = char( bin2dec(num2str(received_chars)))'
```

```
number_of_erroneous_bits =  21
total_number_of_bits =  1050
BER =   0.0200

received_message =Cudio □ateriarking: this messaGe will be embeeded
iN An audio signal( ushnG spread spectrum modulationl and later
qetrimve$ from 4he mo`uleteD!siGnal.
```

System performance overview

A good overview of the system performance can be drawn from the statistical observation of `alpha` values, through a histogram plot. As expected, a bimodal distribution is found, with non zero overlap (Fig. 7.13). Notice that the histogram only gives a rough idea of the underlying distribution, since the number of emitted bits is small.

```
hist(alpha,50);
```

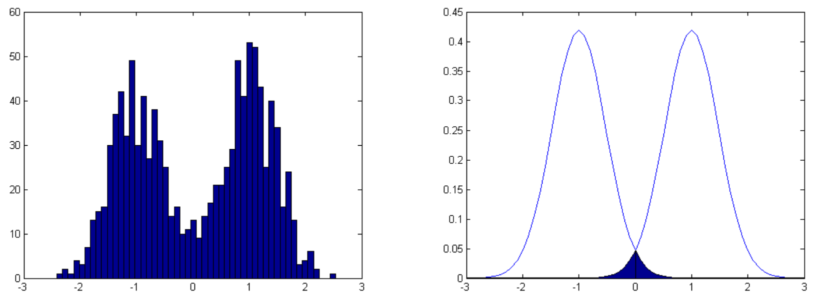

Fig. 7.13 Left: histogram of `alpha` values. Right: theoretical probability density function

In our particular configuration, the spread waveform is a realization of a random sequence with values in {+1,-1}, and the audio-noise vectors are modeled by `N_samples` independent Gaussian random variables with zero mean and variance `sigma2_x`. In this case, it can be shown that the Probability Density Function (PDF) of `alpha` is the average of two normal distributions with mean `a*g` (a=+1 or -1) and variance `sigma2_x/N_samples`. Plotting the theoretical normal distribution shows a better view of the overlap area observed on the histogram. In particular, the area of the overlap between the two Gaussian modes corresponds to the theoretical BER (the exact expression of this BER is given in Section 7.1).

```
alpha_range = -3:.1:3;
gauss0 = 1/2*exp(-((alpha_range-1).^2)/ ...
   (2*sigma2_x/N_samples)) / sqrt(2*pi*sigma2_x/N_samples);
gauss1 = 1/2*exp(-((alpha_range+1).^2)/ ...
   (2*sigma2_x/N_samples)) / sqrt(2*pi*sigma2_x/N_samples);
theoretical_PDF=gauss0+gauss1;

plot(alpha_range, gauss0); hold on
plot(alpha_range, gauss1);

ind0 = find(alpha_range<=0);
ind1 = find(alpha_range>=0);
ae1 = area(alpha_range(ind1), gauss1(ind1), ...
  'FaceColor', [0.39 0.47 0.64]);   hold on;
ae2 = area(alpha_range(ind0), gauss0(ind0), ...
  'FaceColor', [0.39 0.47 0.64]);

theoretical_BER1=2*sum(gauss1(ind1))*(alpha_range(2)-...
   alpha_range(1))
```

```
theoretical_BER1 =  0.0228
```

7.2.2 Informed watermarking with error-free detection

The previous system did not take the audio signal specificities into account (since audio was modeled as white Gaussian noise). We now examine how to modify the system design to reach one major requirement for a watermarking application: that of detecting the watermark message without error. For this purpose, the watermark gain (also called *embedding strength*) has to be adjusted to the local audio variations. This is referred to as informed watermarking.

From now on, the audio signal will be a violin signal sampled at F_s=44.100 Hz, of which only the first `N_bits*N_samples` samples will be watermarked. This signal is normalized in [-1,+1].

```
audio_signal = wavread('violin.wav', [1 N_bits*N_samples]);
```

Informed emitter

To reach a zero BER, the watermark gain is adapted so that the correlation between the watermarked audio signal and the spread sequence is at least equal to some security margin `Delta_g` (if `am` = +1) and `-Delta_g` (if `am` = -1). `Delta_g` sets up a robustness margin against additive perturbation (such as the additive noise introduced by MPEG compression of the watermarked audio signal). In the following MATLAB implementation, `Delta_g` is empirically set to 0.005.

```
Delta_g = 0.005;
for m=0:N_bits-1
  beta = audio_signal(m*N_samples+1:m*N_samples+N_samples)'*...
        spread_waveform /N_samples;
  if symbols(m+1) == 1
        if  beta >= Delta_g
            gain(m+1) = 0;
        else
            gain(m+1) = Delta_g - beta;
        end
  else % if symbols(m+1) == -1
        if  beta <= -Delta_g
            gain(m+1) = 0;
        else
            gain(m+1) = Delta_g + beta;
        end
  end
  watermark_signal(m*N_samples+1:m*N_samples+N_samples,1)= ...

gain(m+1)*modulated_signal(m*N_samples+1:m*N_samples+N_samples);
end

watermarked_signal = watermark_signal + audio_signal;
```

Plotting the gain for one second of signal, together with the resulting watermark signal and the audio signal, shows that the encoder has to make important adjustments as a function of the audio signal (Fig. 7.14).

```
subplot(2,1,1);
plot((0:99)*N_samples/Fs, gain(1:100));
subplot(2,1,2);
plot((0:Fs-1)/Fs, watermark_signal(1:Fs));
```

The watermark, though, is small compared to the audio signal (Fig. 7.15, left).

```
plot((0:Fs-1)/Fs, audio_signal(1:Fs)); hold on;
plot((0:Fs-1)/Fs, watermark_signal(1:Fs), 'r-');
```

Plotting again a few samples of the watermark, audio, and watermarked signals shows that the watermarked signal sometimes differs significantly from the audio signal (Fig. 7.15, right). This is confirmed by a listening test: the watermark is audible.

```
range=(34739:34938);
subplot(3,1,1); plot(range/Fs, watermark_signal(range), 'k-');
subplot(3,1,2); plot(range/Fs, audio_signal(range));
subplot(3,1,3); plot(range/Fs, watermarked_signal(range),'r');
```

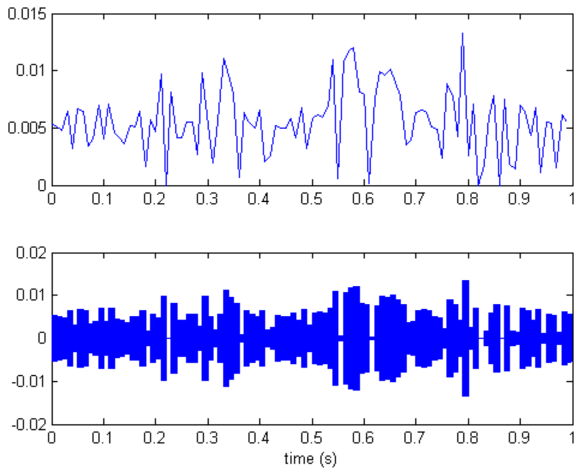

Fig. 7.14 Top: watermark gain; Bottom: watermark signal

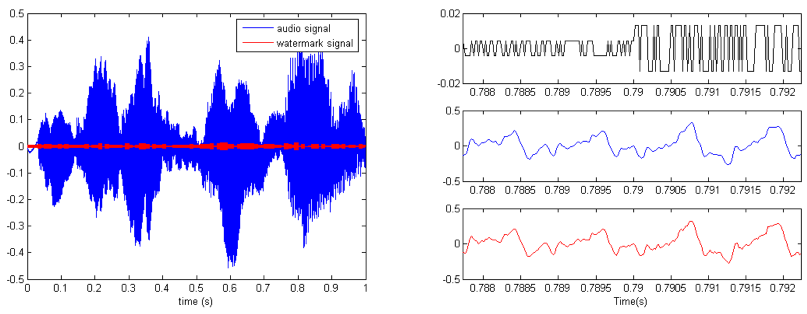

Fig. 7.15 Left: a comparison between the audio and watermark signals; Right: zoom on the watermark signal (top), the audio signal (center), and the watermarked signal (bottom)

Receiver

Using the same correlation demodulator as in Section 7.1, we conclude that the transmission is now effectively error-free (Fig. 7.16), as confirmed by the resulting BER.

```
number_of_erroneous_bits= sum(bits ~= received_bits)
BER = number_of_erroneous_bits/total_number_of_bits
```

```
    received_chars = reshape(received_bits(1:N_bits),N_bits/7, 7);
    received_message = char( bin2dec(num2str(received_chars)))'
```

```
number_of_erroneous_bits =  0
BER =  0

received_message = Audio watermarking: this message will be embedded
in an audio signal, using spread spectrum modulation, and later
retrieved from the modulated signal.
```

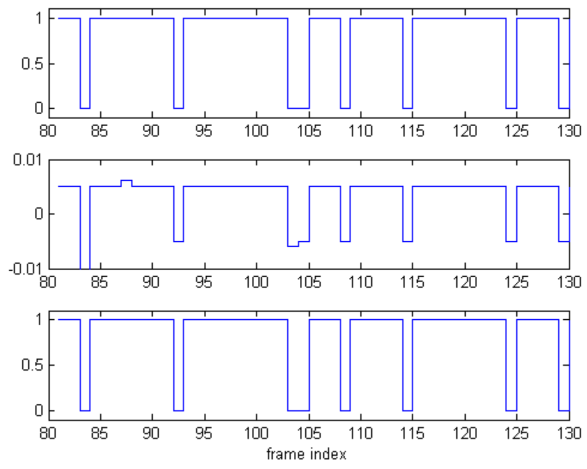

Fig. 7.16 Zoom on bits 80 to 130. Top: input bits; Center: `alpha`; Bottom: received bits

7.2.3 Informed watermarking made inaudible

The informed watermarking system exposed in Section 7.2.2 proves that prior knowledge of the audio signal can be efficiently used in the watermarking process: error-free transmission is reached thanks to an adaptive embedding gain. However, the resulting watermark is audible, since no perceptual condition is imposed on the embedding gain. In this Section, we examine how psychoacoustics can be put to profit to ensure the inaudibility constraint. A psychoacoustic model is used, which provides a signal-dependent masking threshold used as an upper bound for the PSD of the watermark. The watermark gain is therefore replaced by an all-pole perceptual shaping filter and the reception process is composed of a zero-forcing equalizer, followed by a linear-phase Wiener filter.

Emitter, based on perceptual shaping filtering

The perceptual shaping filter is an auto-regressive (all-pole) filter with 50 coefficients `ai` and gain `b0`. It is designed so that the PSD of the watermark (obtained by filtering the `modulated_signal`) equals the `masking_threshold`. The coefficients of the filter are obtained as in Chapter 1, via the Levinson algorithm. Both the masking threshold and the shaping filter have to be updated each time the statistical properties of the `audio_signal` change (here each `N_windows`=512 samples).

Let us first compute the masking threshold and apply the associated shaping filter to one audio frame (the 11-th frame, for instance: Fig. 7.17 left).

```
N_coef = 50;
N_samples_PAM = 512;
PAM_frame = (10*N_samples_PAM+1 :
10*N_samples_PAM+N_samples_PAM);

plot(PAM_frame/Fs,audio_signal(PAM_frame) );
```

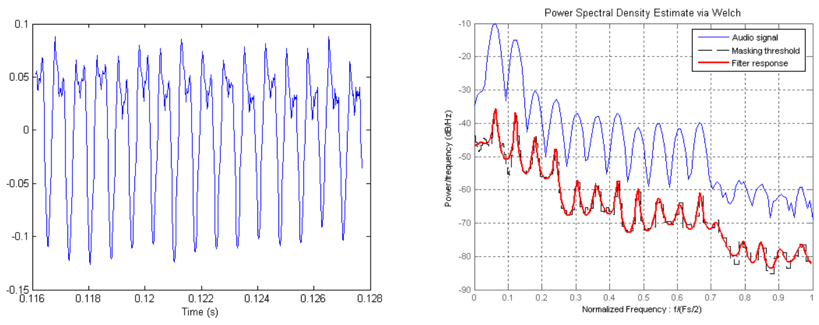

Fig. 7.17 Left: audio signal of frame #11; Right: PSD of frame #11, and the associated masking threshold and shaping filter response

Comparing the periodogram of the audio signal, the masking threshold and the frequency response of the filter shows that the filter closely matches the masking threshold (Fig. 7.17 right). Even the absolute amplitude level of the filter is the same as that of the masking threshold. As a matter of fact, since the audio signal is normalized in [-1,+1], its nominal PSD level is 0 dB.

MATLAB functions involved:

- `masking_threshold=psychoacoustical_model(audio_signal)` returns the masking threshold deduced from a psychoacoustical analysis of the audio vector. This implementation is derived from the psycho-acoustic model #1 used in MPEG-1 Audio (see ISO/CEI norm 11172-3:1993 (F), pp. 122-128 or MATLAB function MPEG1_psycho_acoustic_model1.m from Chapter 3). It is based on the same principles as those used in the MPEG model, but it is further adapted here so as to make it robust to additive noise (which is a specific constraint of watermarking and is not found in MPEG).

- `[b0,ai]=shaping_filter_design(desired_frequency_response_dB,N_coef)` computes the coefficients of an auto-regressive filter:

  $$G(z) = \frac{b0}{ai(1) + ai(2)z^{-1} + \ldots + ai(N\_coef)z^{-N\_coef+1}}$$

  (with ai(1)=1) from the modulus of its `desired_frequency_response` (in dB) and the order `N_coef`. The coefficients are obtained as follows: if zero-mean and unity variance noise is provided at the input of the filter, the PSD of its ouput is given by `desired_frequency_response`. Setting the coefficients so that this PSD best matches `desired_frequency_response` is thus obtained by applying the Levinson algorithm to the autocorrelation coefficients of the output signal (computed itself from the IFFT of the `desired_frequency_response`).

```
masking_threshold = ...
   psychoacoustical_model(audio_signal(PAM_frame));
shaping_filter_response = masking_threshold;
[b0,ai]=shaping_filter_design(shaping_filter_response,N_coef);

% Plotting results. For details on how we use |pwelch|, see
% the appendix of ASP_audio_cd.m, the companion script file of
% Chapter 2.

pwelch(audio_signal(PAM_frame),[],[],[],2);
hold on;
[H,W]=freqz(b0,ai,256);
stairs(W/pi,masking_threshold,'k--');
plot(W/pi,20*log10(abs(H)),'r','linewidth',2);
hold off;
```

Applying this perceptual shaping procedure to the whole watermark signal requires to process the audio signal block per block. Notice that the filtering continuity from one block to another is ensured by the `state` vector, which stores the final state of the filter at the end of one block and applies it as initial conditions for the next block.

```
state = zeros(N_coef, 1);
for m = 0:fix(N_bits*N_samples/N_samples_PAM)-1
    PAM_frame = ...
        (m*N_samples_PAM+1:m*N_samples_PAM+N_samples_PAM);

    % Shaping filter design
    masking_threshold = psychoacoustical_model(...
       audio_signal(PAM_frame) );
    shaping_filter_response = masking_threshold;
    [b0, ai]=shaping_filter_design(shaping_filter_response,...
       N_coef);

    % Filtering stage
    [watermark_signal(PAM_frame,1), state] = ...
        filter(b0, ai, modulated_signal(PAM_frame), state);
end

% Filtering the last, incomplete frame in |watermark_signal|
PAM_frame = ...
    (m*N_samples_PAM+N_samples_PAM+1:N_bits*N_samples);
watermark_signal(PAM_frame,1) = ...
    filter(b0, ai, modulated_signal(PAM_frame), state);

plot((0:Fs-1)/Fs, audio_signal(1:Fs));
hold on;
plot((0:Fs-1)/Fs, watermark_signal(1:Fs), 'r-');
```

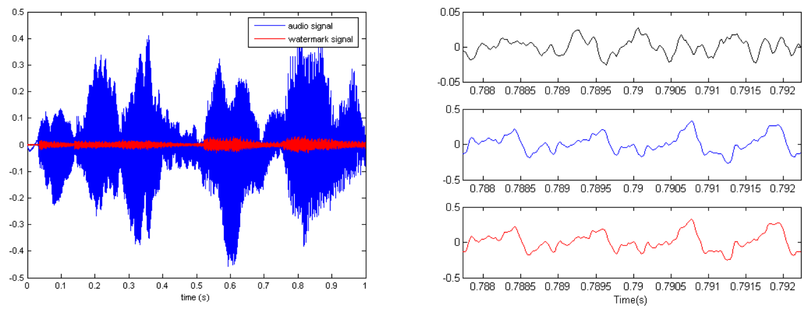

Fig. 7.18 Left: audio signal and watermark signal; Right: a few samples of the audio, watermark, and watermarked signals

The watermarked audio signal is still obtained by adding the audio signal and the watermark signal (Fig. 7.18 left). Plotting again few samples of the watermark, audio, and watermarked signals shows that the watermark signal has now been filtered (Fig. 7.18 right). As a result, the watermark is quite inaudible, as confirmed by a listening test. Its level, though, is similar to (if not higher than) that of the watermark signal in Section 7.2.2.

```
watermarked_signal = watermark_signal + audio_signal;

range=(34739:34938);
subplot(3,1,1); plot(range/Fs, watermark_signal(range), 'k');
subplot(3,1,2); plot(range/Fs, audio_signal(range));
subplot(3,1,3); plot(range/Fs, watermarked_signal(range),'r');
```

Receiver, based on zero-forcing equalization and detection

The zero-forcing equalization aims at reversing the watermark perceptual shaping, before extracting the embedded message. The shaping filter with frequency response shaping_filter_response is therefore recomputed from the audio watermarked_signal, since the original audio_signal is not available at the receiver. This process follows the same block processing as the watermark synthesis, but involves a moving average filtering stage: filtering the watermarked_signal by the filter whose frequency response is the inverse of the shaping_filter_response yields the filtered received signal denoted by equalized_signal.

Plotting the PSD of the equalized_signal shows its flat spectral envelope, hence its name (Fig. 7.19).

```
state = zeros(N_coef, 1);
for m = 0:fix(N_bits*N_samples/N_samples_PAM)-1

    PAM_frame = ...
       m*N_samples_PAM+1:m*N_samples_PAM+N_samples_PAM;

    % Shaping filter design, based the watermarked signal
    masking_threshold =
       psychoacoustical_model(watermarked_signal(PAM_frame));
    shaping_filter_response = masking_threshold;
    [b0, ai]=shaping_filter_design(shaping_filter_response,...
       N_coef);

    % Filtering stage
    [equalized_signal(PAM_frame), state] = ...
        filter(ai./b0,1,watermarked_signal(PAM_frame),state);

    if m==10
    % Showing the frequency response of the equalizer and PSD
    % of the |equalized_signal|, for frame 10.
        pwelch(equalized_signal(PAM_frame),[],[],[],2);
        hold on;
        [H,W]=freqz(ai./b0,1,256);
        plot(W/pi,20*log10(abs(H)),'r','linewidth',2);
    end;

end
```

The equalized_signal is theoretically the sum of the original watermark (i.e. the modulated_signal, which has values in {-1,+1}) and the equalized_audio_signal, which is itself the original audio_signal filtered by the Zero-Forcing equalizer. In other words, from the receiver point of view, everything looks as if the watermark had been added to the equalized_audio_signal rather than to the audio_signal itself.

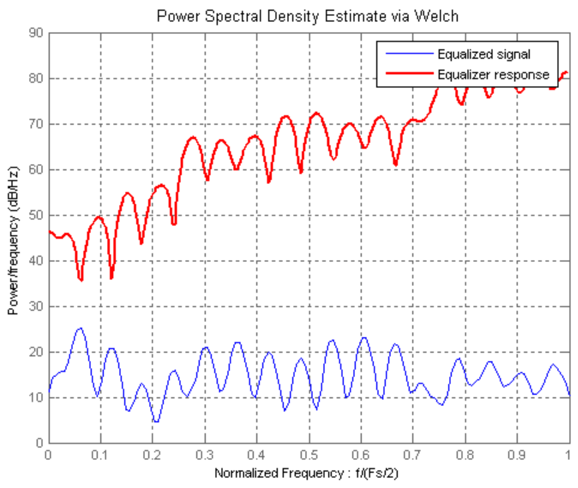

Fig. 7.19 Frequency response of the equalizer at frame #10, and PSD of the resulting equalized watermarked signal

Although the `equalized_audio_signal` is not available to the receiver, it is interesting to compute it and compare it to the original `audio_signal` (Fig. 7.20, left). Notice that the level of the `equalized_audio_signal` is much higher than that of the original `audio_signal`, mostly because its HF content has been enhanced by the equalizer.

```
equalized_audio_signal = zeros(N_bits*N_samples, 1);
state = zeros(N_coef, 1);

for m = 0:fix(N_bits*N_samples/N_samples_PAM)-1

    PAM_frame = ...
       m*N_samples_PAM+1:m*N_samples_PAM+N_samples_PAM;

    % Shaping filter design, based the watermarked signal
    masking_threshold = ...
       psychoacoustical_model(watermarked_signal(PAM_frame));
    shaping_filter_response = masking_threshold;
    [b0,ai]=shaping_filter_design(shaping_filter_response, ...
       N_coef);

    % Filtering stage
    [equalized_audio_signal(PAM_frame), state] = ...
        filter(ai./b0, 1, audio_signal(PAM_frame), state);
end

% Plotting 200 samples of the |equalized_audio_signal|, and
% |modulated_signal|
range=(34739:34938);
subplot(2,1,1); plot(range/Fs, equalized_audio_signal(range));
subplot(2,1,2); plot(range/Fs, modulated_signal(range));
```

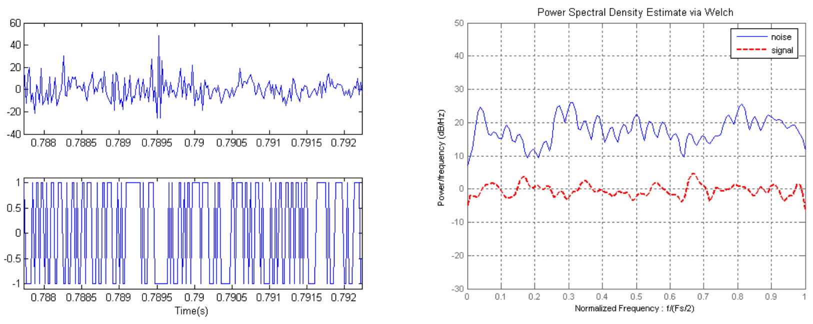

Fig. 7.20 Left: zoom on the equalized audio signal and on the modulated signal; Right: corresponding PSDs

It is also possible to estimate the SNR by visual inspection of the PSDs of the equalized audio signal and the modulated signal (Fig. 7.20, right) or to compute it from the samples of these signals.

```
range=(34398:34938);
pwelch(equalized_audio_signal(range),[],[],[],2);
[h,w]=pwelch(modulated_signal(range),[],[],[],2);
hold on;
plot(w,10*log10(h),'--r ');

snr_equalized=10*log10(var(modulated_signal(range))/...
   var(equalized_audio_signal(range)))
```

```
snr_equalized =  -19.0610
```

We can now proceed to the watermark extraction by applying the correlation demodulator to the `equalized_signal`, and compute the BER. As expected, the obtained BER is quite high, since the SNR is low: the level of the `equalized_audio_signal` is high compared to that of the embedded `modulated_signal` (Fig. 7.20).

```
for m = 0:N_bits-1
    alpha(m+1) = (equalized_signal(m*N_samples+1: ...
        m*N_samples+N_samples)' * spread_waveform)/N_samples;
    if alpha(m+1) <= 0
        received_bits(m+1) = 0;
    else
        received_bits(m+1) = 1;
    end
end

% Plotting the input bits, alpha, and the received bits.
range=(81:130);
subplot(3,1,1);
stairs(range, bits(range));
subplot(3,1,2);
stairs(range, alpha(range));
subplot(3,1,3);
```

```
stairs(range, received_bits(range));

number_of_erroneous_bits= sum(bits ~= received_bits)
total_number_of_bits=N_bits
BER = number_of_erroneous_bits/total_number_of_bits

received_chars = reshape(received_bits(1:N_bits),N_bits/7, 7);
received_message = char( bin2dec(num2str(received_chars)))'
```

```
number_of_erroneous_bits =    30
total_number_of_bits =   1050
BER =  0.0286

received_message = udio watdrmarkyng:0plks mussage will be embedded
in e~0audik siwnql, }wi~g spread SpeCtruM(modul`tion, and
later0retrieved from the modudated qienal.
```

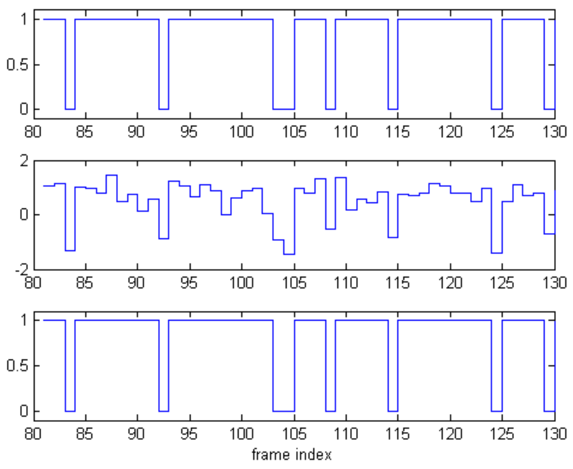

Fig. 7.21 Zoom on bits 80 to 130. Top: input bits; Center: `alpha`; Bottom: received bits

Wiener filtering

The Wiener filtering stage aims at enhancing the SNR between the `modulated_signal` and the `equalized_audio_signal`. This is achieved here by filtering the `equalized_signal` by a symmetric (non causal) FIR filter with `N_coef`=50 coefficients. Its coefficients `hi` are computed so that the output of the filter (when fed with the `equalized_signal`) becomes maximally similar to the `modulated_signal` in the RMSE sense. They are the solution of the so-called Wiener-Hopf equations:

$$\texttt{hi} = \texttt{equalized\_signal\_cov\_mat}^{-1} * \texttt{modulated\_signal\_autocor\_vect}$$

where `equalized_signal_cov_mat` and `modulated_signal_autocor_vect` are the covariance matrix of the `equalized_signal` and the autocorrelation vector of the `modulated_signal`, respectively.

Since the `modulated_signal` is unknown from the receiver, its autocorrelation is estimated from an arbitrary modulated signal and can be computed once. To make it simple, we use our previously computed `modulated_signal` here. On the contrary, the covariance matrix of the equalized signal, and therefore the coefficients `hi`, has to be updated each time the properties of `estimated_signal` change, that is every `N_samples_PAM`=512 samples. Wiener filtering is then carried out, by computing the covariance matrix of the `equalized_signal` and the impulse response of the Wiener filter for each PAM frame.

```
modulated_signal_autocor_vect=...
   xcorr(modulated_signal,N_coef,'biased');

state = zeros(2*N_coef, 1);

for m = 0:fix(N_bits*N_samples/N_samples_PAM)-1
    PAM_frame = ...
       (m*N_samples_PAM+1:m*N_samples_PAM+N_samples_PAM);

    % Estimating the covariance matrix of |equalized signal|,
    % as a Toeplitz matrix with first row given by the
    % autocorrelation vector of the signal.
    equalized_signal_autocor_vect= ...
        xcorr(equalized_signal(PAM_frame),2*N_coef,'biased');
    equalized_signal_cov_mat= ...
      toeplitz(equalized_signal_autocor_vect(2*N_coef+1:end));

    % Estimating the impulse response of the Wiener filter as
    % the solution of the Wiener-Hopf equations.
    hi=equalized_signal_cov_mat\modulated_signal_autocor_vect;

    % Filtering stage
    [Wiener_output_signal(PAM_frame,1), state] = ...
        filter(hi, 1, equalized_signal(PAM_frame), state);
    power = ...
        norm(Wiener_output_signal(PAM_frame))^2/N_samples_PAM;
    if (power ~= 0)
        Wiener_output_signal(PAM_frame) = ...
           Wiener_output_signal(PAM_frame)/ sqrt(power);
    end

    % Saving the Wiener filter for frame #78
    if m==78
        hi_78=hi/sqrt(power);
    end;

end

% Since the Wiener filter is non-causal (with |N_coeff|=50
% coefficients for the non-causal part), the resulting
% |Wiener_output_signal| is delayed with |N_coef| samples.
Wiener_output_signal = ...
   [Wiener_output_signal(N_coef+1:end);  zeros(N_coef, 1)];
```

It is interesting to check how the equalized audio signal and the modulated signal have been modified by the Wiener filter (Fig. 7.22 left) and how their PSDs have evolved (Fig. 7.22 right). Obviously, the Wiener filter has enhanced the frequency bands dominated by the modulated signal in Fig. 7.20 (right), thereby increasing the SNR by more than 5 dBs.

```
range=(34398:34938);
Wiener_output_audio=filter(hi,1,equalized_audio_signal(range));
Wiener_output_modulated=filter(hi,1,modulated_signal(range));
range=(34739:34938);
subplot(2,1,1); plot(range/Fs,Wiener_output_audio(341:540));
subplot(2,1,2); plot(range/Fs,Wiener_output_modulated(341:540));

pwelch(Wiener_output_audio,[],[],[],2);
[h,w]=pwelch(Wiener_output_modulated,[],[],[],2);
hold on;
plot(w,10*log10(h),'--r ', 'linewidth',2);

snr_Wiener=10*log10(var(Wiener_output_modulated(341:530)) ...
      /var(Wiener_output_audio(341:530)))
```

```
snr_Wiener =  -19.2023
```

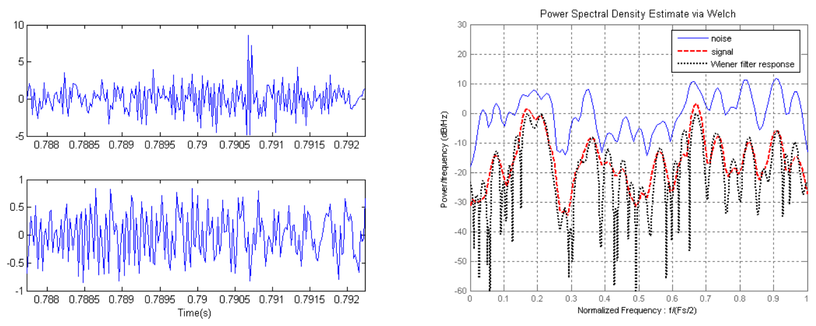

Fig. 7.22 Left: zoom on the equalized audio signal and on the modulated signal, passed through the Wiener filter; Right: corresponding PSDs, and frequency response of the Wiener filter.

We can finally apply the correlation demodulator to the estimated modulated signal. The resulting BER is lower (Fig. 7.23), thanks to Wiener filtering: a BER of the order of 10^{-3} is reached, while the watermark is now inaudible.

```
for m = 0:N_bits-1
    alpha(m+1) = (Wiener_output_signal(m*N_samples+1: ...
        m*N_samples+N_samples)'*spread_waveform )/N_samples;
    if alpha(m+1) <= 0
        received_bits(m+1) = 0;
    else
```

```
            received_bits(m+1) = 1;
        end
    end

    number_of_erroneous_bits= sum(bits ~= received_bits)
    total_number_of_bits=N_bits
    BER = number_of_erroneous_bits/total_number_of_bits

    received_chars = reshape(received_bits(1:N_bits),N_bits/7, 7);
    received_message = char( bin2dec(num2str(received_chars)))'
```

```
number_of_erroneous_bits =  2
total_number_of_bits =   1050
BER =   0.0019

received_message = Audio watermarking: this message will$be embedded
in an audIo signal, using spread spectrum modulation, and later
retrieved from the modulated signal.
```

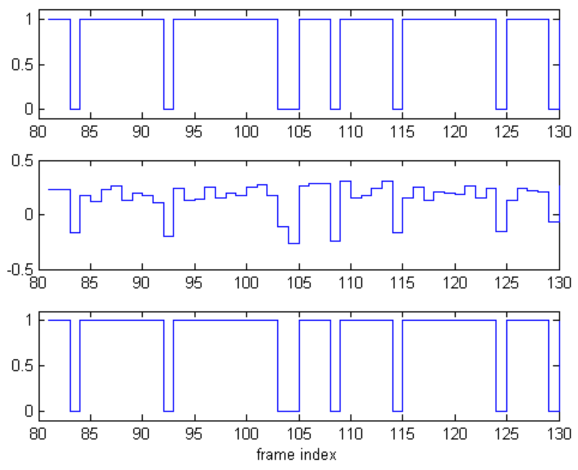

Fig. 7.23 Zoom on bits 80 to 130. Top: input bits; Center: `alpha`; Bottom: received bits

Robustness to MPEG compression

We finally focus on the robustness of our system to MPEG compression. Using the MATLAB functions developed in Chapter 3, we can easily apply an mp3 coding/decoding operation to the audio watermarked signal, yielding the distorted watermarked audio signal `compressed_watermarked_signal`.

MATLAB function involved:

- `output_signal = codec_mp3( input_signal, Fs)` returns the signal `output_signal` resulting from a mp3 coding/decoding operation of the input signal `input_signal` sampled at the `Fs` frequency sampling (see chapter 3).

```
compressed_watermarked_signal=mp3_codec(watermarked_signal,Fs);
```

We then pass the compressed signal through the Zero-Forcing equalizer and the Wiener filter, as before, and compute the resulting BER [13] and received message.

```
BER =    0.1048
received_message = )-%im$gate□mapcHng□"txIr }!UsagM
wiLm&bD$embed`de□hN"cn yUdho skenan, Usino`sppdaD sp}btrum
mOd5latio~( a^E□Lat%□02gtrievgd fPoMda*uOm/$uXetet4cggna}
```

Unfortunately, the obtained BER has significantly increased: although it has been shown above that the psychoacoustic model we use in our perceptual watermarking system is robust to the watermark (i.e., the masking threshold does not change significantly when the watermark is added to the audio signal), it is clearly not robust *yet* to MPEG compression.

To show this, let us compare the PSD of the watermarked signal and the associated masking threshold, before and after the MPEG compression, on the 11th frame for instance (Fig. 7.24). Clearly, the masking threshold is very sensitive to MPEG compression for frequencies over 11 kHz. As a result, the perceptual shaping filter and the equalizer no longer cancel each other, hence our high BER.

```
PAM_frame = (10*N_samples_PAM+1 : ...
   10*N_samples_PAM+N_samples_PAM);

subplot(1,2,1)
pwelch(watermarked_signal(PAM_frame),[],[],[],2);
hold on;
masking_threshold = psychoacoustical_model(...
   watermarked_signal(PAM_frame) );
stairs((1:N_samples_PAM/2)*2/N_samples_PAM,masking_threshold,...
   'r','linewidth',2)

subplot(1,2,2)
pwelch(compressed_watermarked_signal(PAM_frame),[],[],[],2);
hold on;
masking_threshold = psychoacoustical_model(...
   compressed_watermarked_signal(PAM_frame) );
stairs((1:N_samples_PAM/2)*2/N_samples_PAM,masking_threshold,...
   'r','linewidth',2)
```

[13] The corresponding MATLAB code can be found in the `ASP_watermarking.m` file. We do not repeat it here.

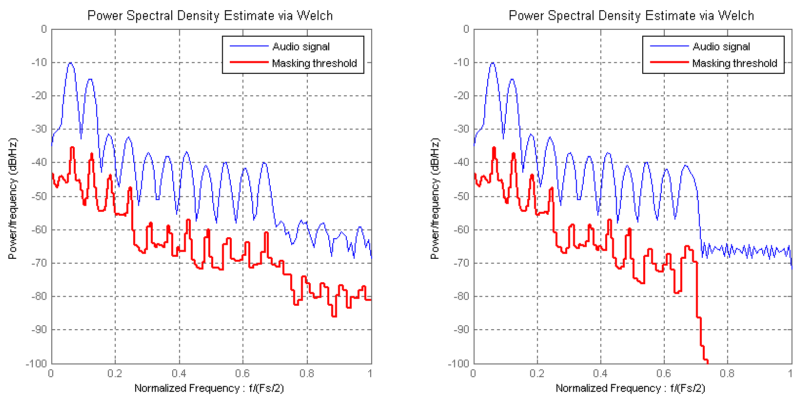

Fig. 7.24 PSD of frame #11, and the associated masking threshold. Left: before MPEG compression; Right: after MPEG compression

7.2.4 Informed watermarking, robust to MPEG compression

In order to improve the robustness of the system exposed in the previous Section, the watermark information should obviously be spread in the [0 Hz, 11 kHz] frequency range of the audio signal. This can be achieved with a low pass filter with cutoff frequency set to 11 kHz (Baras *et al.* 2006). As we shall see below, this filter will interfere with our watermarking system in three stages.

Designing the low pass filter

First, we design a symmetric FIR low pass filter with `Fc=11` kHz cutoff frequency, using the Parks-McClellan algorithm. Such a symmetric filter (Fig. 7.25, left) will have linear phase, and therefore will not change the shape of the modulated signal more than required. We set the order of the filter to `N_coef`=50. Notice that this filter is non-causal and introduces a `N_coef`/2-samples delay.

```
Fc = 11000;
low_pass_filter = firpm(N_coef, [0 Fc-1000 Fc+1000 Fs/2]*2/Fs,
   [1 1 1E-9 1E-9]);
%low_pass_filter = ...
   low_pass_filter/sqrt(sum(low_pass_filter)^2));

% Let us plot the impulse response of this filter.
plot(low_pass_filter);
```

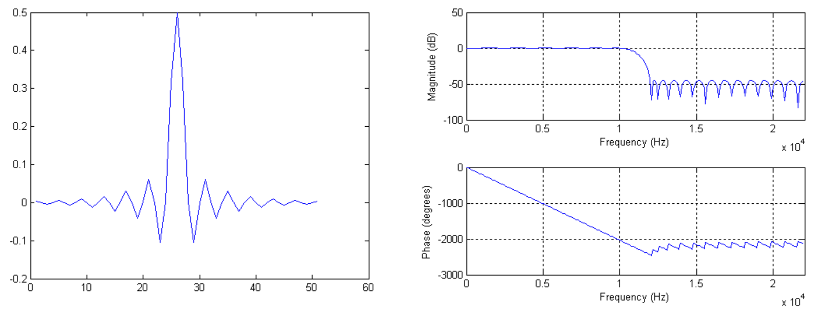

Fig. 7.25 Left: Impulse response of a symmetric FIR low-pass filter; Right: Its frequency response

It is easy to check that its frequency response matches our requirements (Fig. 7.25, right).

```
freqz(low_pass_filter,1,256);
```

Modifying the emitter

We first use this low-pass filter to create a spread sequence with spectral content restricted to the [0 Hz, 11 kHz] range. The modulated signal can then be designed as previously.

```
spread_waveform = ...
   filter(low_pass_filter,1,2*round(rand(N_samples+N_coef,1))-1);
% Getting rid of the transient
spread_waveform = spread_waveform(N_coef+1:end);
% Making sure the norm of the spread waveform is set to
% sqrt(N_samples), as in the previous Sections.
spread_waveform = spread_waveform/ ...
    sqrt(norm(spread_waveform).^2/N_samples);

for m = 0:N_bits-1
    modulated_signal(m*N_samples+1:m*N_samples+N_samples) = ...
        symbols(m+1)*spread_waveform;
end
```

To prevent the shaping filter from amplifying the residual frequency component of the modulated signal over 11 kHz, the `shaping_filter_response`, initially chosen to be equal to the masking threshold, is now designed to be equal to the masking threshold in the [0, 11] kHz frequency band and to zero anywhere else.

```
state = zeros(N_coef, 1);
N_cutoff = ceil(Fc/(Fs/2)*(N_samples_PAM/2));

for m = 0:fix(N_bits*N_samples/N_samples_PAM)-1
    PAM_frame = ...
       (m*N_samples_PAM+1:m*N_samples_PAM+N_samples_PAM);

    % Shaping filter design
    masking_threshold = ...
        psychoacoustical_model( audio_signal(PAM_frame) );
    shaping_filter_response = [masking_threshold(1:N_cutoff); ...
        -100*ones(N_samples_PAM/2-N_cutoff, 1) ];
   [b0,ai]=shaping_filter_design(shaping_filter_response,N_coef);

    % Filtering stage
    [watermark_signal(PAM_frame), state] = ...
        filter(b0, ai, modulated_signal(PAM_frame), state);
end

watermarked_signal = audio_signal + watermark_signal;
```

MPEG compression is then applied to the new `watermarked_signal`.

```
compressed_watermarked_signal=mp3_codec( watermarked_signal,Fs);
```

Modifying the receiver

Before starting the watermark extraction, the `compressed_watermarked_signal` is low-passed, to avoid residual components over 11 kHz.

```
compressed_watermarked_signal = ...
   filter(low_pass_filter, 1, [compressed_watermarked_signal; ...
      zeros(N_coef/2, 1)]);
% Compensating tof the |N_coef/2| delay
compressed_watermarked_signal = ...
    compressed_watermarked_signal(N_coef/2+1:end);
```

Watermark extraction is finally obtained as in Section 7.2.3, except the LP filter is again taken into account in the zero-forcing equalization[14]. The resulting BER is similar to the one we obtained with no MPEG compression: our watermarking system is thus now robust to MPEG compression.

```
BER =    0.00095

received_message = A5dio watermarking: this message will be embedded
in an audio signal, using spread spectrum modulation, and later
retrieved from the modulated signal.
```

[14] The corresponding MATLAB code can be easily found in the `ASP_watermarking.m` file. We do not repeat it here.

7.3 Going further

Readers interested in a more in-depth study of spread spectrum communications will find many details, as well as MATLAB examples, in (Proakis *et al.* 2000).

Several books are available on digital watermarking techniques. (Cox *et al,* 2001) provides an intuitive approach, and many examples in C. Source code can also be found in (Pan *et al.* 2004).

Notice also that this Chapter has only mentioned *additive watermarking* (yet only in the time-domain), as opposed to the other main approach, known as *substitutive watermarking* (Arnold *et al.* 2003). Moreover, additive watermarking can be performed in various domains, leading to various bit-rate vs. distorsion vs. robustness trade-offs (Cox *et al.* 2001): in the time-domain, in the frequency-domain, in the amplitude or phase domain (such as the one corresponding to the output of a Modulated Lapped Transform, for robustness to small de-synchronization; see Malvar and Florencio, 2003), in the cepstral domain (often used in speech processing for its relation to the source/filter model; see Lee and Ho, 2000), in other parametric domains (such as the MPEG compressed domain; see Siebenhaar *et al.* 2001). Last but not least, we have not examined synchronization problems here, which require special attention in the context of spread spectrum communications, since even a one-sample delay will have a disastrous effect on the resulting BER (Baras *et al.* 2006).

Security is one of the emerging challenges in watermarking (Furon *et al.* 2007). For more challenges, see for instance the BOWS challenges organized by the within European Network of Excellence ECRYPT (Bows2, 2007).

7.4 Conclusion

Hiding bits of information in an audio stream is not as hard as one might think. It can be achieved by modulating the transmitted bits by a spread spectrum sequence, and adding the resulting watermark signal to the audio stream. Making the watermark inaudible is a matter of dynamically shaping its spectrum so that it falls below the local masking threshold provided by a psycho-acoustical model. Last but not least, ensuring that the system is robust to MPEG compression simply requires the addition of a low-pass filter in the spectrum-shaping stage.

Provided the receiver is able to invert the spectrum-shaping operation using the same psycho-acoustical model as the emitter, and with the additional help of a Wiener filter to increase the SNR, a bit error rate of the order of 0.001 can be obtained.

7.5 References

Arnold M, Wolthusen S, Schmucker M (2003) Techniques and applications of digital water-marking and content protection. Artech House Publishers, Norwood, MA

Bows2 (2007) Break our Watermarking system, 2nd edition [online] Available: http://bows2.gipsa-lab.inpg.fr [02/1/2008]

Baras C, Moreau N, Dymarski P (2006) Controlling the inaudibility and maximizing the robustness in an audio annotation watermarking system. IEEE Transactions on Audio, Speech and Language Processing, 14-5:1772–1782

Chen B, Wornell G (2001) Quantization index modulation: a class of provably good methods for digital watermarking and information embedding. IEEE Transactions on Information Theory, 47:1423–1443

Costa M (1983) Writing on dirty paper. IEEE Transactions on Information Theory, 29:439–441

Cox I, Miller M, Bloom J (2001) Digital Watermarking: Principles and Practice Morgan Kaufmann, San Francisco, CA

Cox I, Miller M, McKellips A (1999) Watermarking as communications with side information. Proceedings of the IEEE, 87–7: 1127–1141

Craver SA, Wu M, Liu B (2001) What Can We Reasonably Expect From Watermarks? Proceedings of the IEEE Workshop on Applications of Signal Processing to Audio and Acoustics, New Paltz, NY, 223–226

Furon T, Cayre F, Fontaine C (2007) Watermarking Security. In: Cvejic and Seppanen (eds) Digital Audio Watermarking Techniques and Technologies: Applications and Benchmarks. Information Science Reference, Hershey, PA (USA)

Haykin S (1996) Adaptive filter theory, 3rd ed. Prentice-Hall, Upper Saddle River,NJ (USA)

Jobs S (2007) Thoughts on Music [online] Available: http://www.apple.com/hotnews/thoughtsonmusic/ [26/9/2007]

Kirovski D, Malvar H (2003) Spread-spectrum watermarking of audio signals. IEEE Transactions on Signal Processing, 51-4:1020–1033

Larbi S, Jaïdane M, Moreau N (2004). A new Wiener filtering based detection scheme for time domain perceptual audio watermarking. In IEEE Interna tional Conference on Acoustics, Speech and Signal Processing (ICASSP), Montreal, Canada, 5:949–952

Lee SK, Ho YS (2000) Digital audio watermarking in the cepstrum domain. IEEE Transactions on Consumer Electronics, 46-3: 744–750.

LoboGuerrero A, Bas P, Lienard J (2003) Iterative informed audio data hiding scheme using optimal filter," in Proc. IEEE Int. Conf. on Communication Technology, Beijing, China, 1408–1411

Malvar H, Florencio D (2003) "Improved spread spectrum: a new modulation technique for robust watermarking" IEEE Transactions on Signal Processing, 51-4:898–905

Massey JL (1994) Information theory aspects of spread-spectrum communications. IEEE Third International Symposium on Spread Spectrum Techniques and Applications (ISSSTA), 1:16–21

Pan JS, Huang HC, Jain LC (2004) Intelligent Watermarking Techniques (Innovative Intelligence). World Scientific Publishing Company, Singapore

Petitcolas FAP, Anderson RJ, Kuhn MG (1999) Information hiding – a survey. Proceedings IEEE, special issue on protection of multimedia content, 87(7):1062–1078

Peterson RL, Ziemer RA, Borth DE (1995) Introduction to spread spectrum communications. Prentice Hall, Upper Saddle River, NJ (USA)

Proakis JG, Salehi M, Bauch G (2004) Contemporary Communication Systems Using MATLAB and SIMULINK. Brooks/Cole-Thomson Learning, Belmont, CA (USA)

Shannon C (1958) Channel with side information at the transmitter. IBM Journal of Research and Development, 2: 222–293

Siebenhaar F, Neubauer C, Herre J, Kulessa R (2001) New results on combined audio compression/watermarking. In: 111th Convention of Audio Engineering Society (AES), New York, USA, preprint 5442.

Proakis J (2001) Digital communications, 4th ed. New York, USA: McGraw-Hill