

Digital Audio Coding

Lab Session 4 – AWARE

Sébastien Boisgérault,
Mines-ParisTech.

March 20, 2013
CC BY 3.0 License.

This lab session is dedicated to the implementation of a perceptual audio coder using filter banks, psychoacoustic models and vector quantization of subband data. For the sake of simplicity, we will only consider stationary sounds so that we may compute the psychoacoustic mask of the sound only once.

In this lab session, the sampling frequency Δf is 44.1 kHz. Spectral analysis will be performed on frames of 512 samples and vector quantization will be applied on frames of 12 samples in each of the 32 subbands of the filter banks.

You are advised to deal first with the questions identified with a † symbol.

Filter Banks

1. **Analysis Filter Bank.** The code

```
>>> analyze = Analyzer(MPEG.A, dt=MPEG.dt)
```

creates an instance of an analysis filter bank based on of the pseudo-quadrature mirror filters (PQMF) of the MPEG standard. This polyphase implementation consumes frames of 32 successive samples and outputs an arrays of 32 values, one for each subband.

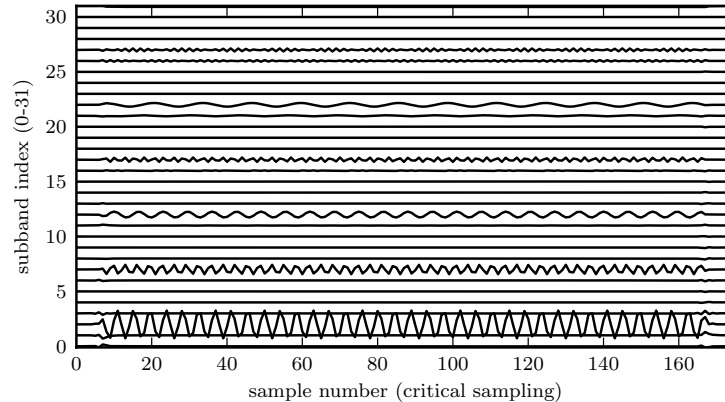
```
>>> assert shape(frames) == (32,)
>>> subband_data = analyze(frame)
```

Use this filter bank to implement `display_subbands`, a function that displays graphically the subband decomposition of signals of audio data of arbitrary length.

† Test this function with the following signals:

```
>>> data_sin = tone(f=1760.0, N=10000)
>>> data_squ = square(f=1760.0, N=10000)
>>> data_noi = white_noise(N=10000)
```

Subband decomposition of a square signal (unit amplitude, $f = 1760$ Hz).



2. **Synthesis Filter Bank.** Symmetrically, a PQMF synthesis filter bank may be instantiated with:

```
>>> synthesizer = Synthesizer(MPEG.S, dt=MPEG.dt, gain=MPEG.M)
```

The application of this instance to subband data generates audio frames:

```
>>> frame = synthesizer(subband_data)
```

† Use a test function – such as an impulse – to determine how much delay the whole analysis and synthesis process induces.

Implement a function `reconstruct` that will take an array of samples for argument and return an array of the same size that has been through analysis and synthesis and for which the delay has been compensated.

† Is the reconstruction perfect ? Determine experimentally the order of magnitude of the signal-to-noise of the reconstruction process ? Is it good enough ?

Psychoacoustics Mask

Let I be the (normalized) intensity of the frame x :

$$I = \langle x^2(t) \rangle = \frac{1}{512} \sum_{n=0}^{511} x(n\Delta t)^2 \quad (1)$$

1. **Power Spectrum.** The sound intensity may be computed in the frequency domain as

$$I = \frac{\Delta f}{256} \int_0^{\Delta f/2} |x(f)|^2 df.$$

Consequently, if $f_k = k\Delta f/512$ and \hat{x}_k is the Discrete Fourier Transform of a frame x with 512 values, the frequency f_k for $k = 0, \dots, 257$ contributes I_k to the signal intensity with $I_k \simeq 2(|\hat{x}_k|/512)^2$ if $0 < f_k < \Delta f/2$ and $I_k \simeq (|\hat{x}_k|/512)^2$ if $f_k = 0$ or $f_k = \Delta f/2$.

Implement a function `Ik` that uses the Fast Fourier Transform to compute the array of values I_k when the argument `x` is a frame of 512 sample values.

† Check with a random frame `x` that the sum of the I_k is equal to the sound intensity I .

Introduce an optional argument `dB` that defaults to `True` to compute the intensities as sound pressure levels, in dB, by default (when `dB` is `False`, intensities are returned in a linear scale).

† Display the power spectrum I_k of the pure tone A8 whose frequency is 7040 Hz as a function of the frequency f_k .

Extend the function `Ik` with an optional `window` argument, that will be applied to the array `x` before the spectral analysis. As this process will likely alter the signal intensity, the array `x` should also be scaled to attempt to restore the initial intensity level.

† Create a frame made of a pure tone with frequency 7040 Hz (A₈) and another one with frequency 14080 Hz (A₉) whose amplitude is one hundredth (1/100) of the first. Display the power spectrum of a this frame `x` with and without a hanning window. What is the purpose of the window in this context ?

2. **Tonal/Non-Tonal Classification.** The component $k \in \{3, \dots, 249\}$ of the array I_k is considered tonal if I_k is greater than or equal to I_{k-1} and PI_{k+1} and $I_k \geq I_{k+j} + 7.0$ dB for every $j \in J_k$.

| k | J_k |
|--------------------|--|
| $003 \leq k < 063$ | $\{-2, +2\}$ |
| $063 \leq k < 127$ | $\{-3, -2, +2, +3\}$ |
| $127 \leq k < 250$ | $\{-6, -5, -4, -3, -2, +2, +3, +4, +5, +6\}$ |

Implement a function `sort_maskers` that given an intensity array `Ik` in dB of length 257 returns two arrays, `tonal` and `non_tonal` of the same

size. The first one should contain the the tonal intensity components of I_k and their immediate neighbours – or $-\infty$ otherwise – and the second one the non-tonal components – or $-\infty$ otherwise.

3. **Masking Patterns.** We will assume that a single masker with frequency b_m barks and sound pressure level I_m (in dB) generates at the frequency b barks a mask level of

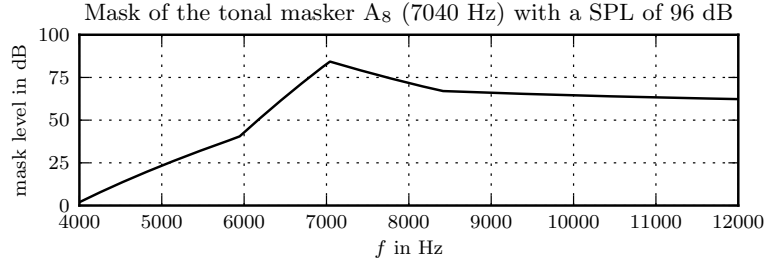
$$\text{mask level [dB]} = m(b) + a(b) \quad \text{with} \quad \Delta b = b - b_m \quad (2)$$

where the base mask level $m(b)$ is defined as

$$m(b) = \begin{cases} (+11.0 - 0.40 \times I_m) \times (\Delta b + 1.0) & \text{if } \Delta b < -1.0 \\ + (+6.0 + 0.40 \times I_m) \times (\Delta b + 0.0) & \text{if } \Delta b < 0.0 \\ + (-17.0) \times (\Delta b + 0.0) & \text{if } 0.0 \leq \Delta b \\ + (0.15 \times I_m) \times (\Delta b - 1.0) & \text{if } 1.0 \leq \Delta b \end{cases}$$

and the attenuation $a(b)$ is given by

$$a(b) = \begin{cases} -1.525 - 0.275 \times b - 4.5 & \text{if the masker is tonal,} \\ -1.525 - 0.175 \times b - 0.5 & \text{otherwise.} \end{cases}$$



† In this model, is the masking effect of tonal sounds stronger or weaker than the one of non-tonal sounds ? Is a loud voice with a high pitch more likely to mask another loud voice with a low pitch or the opposite ?

Implement a function `excitation_pattern` with arguments `f` and `I` the frequency of the masker in Hz and the intensity in dB and a boolean argument `tonal`, that returns the mask level as a function of the frequency in Hz.

4. **Composite Masks.** Implement a function `mask_from_frame` that computes the power spectrum of a frame, the mask associated to every spectral component and creates a global mask function from the addition of the all mask intensities¹. Add an optional argument `floor` that defaults to `ATH`, the absolute threshold of hearing, defined in the `psychoacoustics` module, that will be added as an extra mask component if it is not `None`.

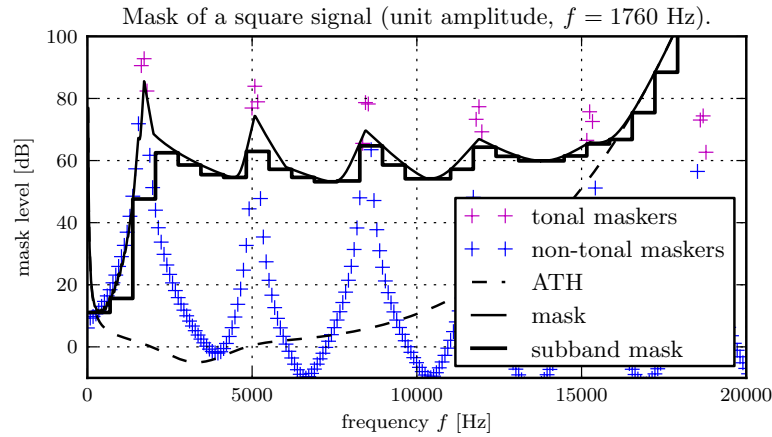
¹**Warning:** the addition should *not* be performed in dB but in the original intensity units. The `Mask` compositor from the `psychoacoustics` module may be used to perform the necessary conversions.

5. **Sampling Masks on Subbands.** In the previous section, we have decomposed the sound data into 32 subbands of $[0, \Delta f/2]$ with the same width. We will require in next section an estimation of the mask level in each subband.

† If we take for subband mask level the value of the mask level in the middle of the band, and pick a pure tone in the subband with an intensity below that threshold, are we sure that the pure tone is not audible? How should we select the subband mask level if we want to be more conservative?

Design a function `sample_mask` – whose only argument is the mask function – that implements such as pessimistic sampling of the mask function.

6. **Tests.** † Apply the previous computations with frames of stationary signals such as pure tones, white noise, square signals, etc. and display graphically the results in a function `display_mask`.



Subband Data Vector Quantization

1. **Scale Factors.** The quantization of subband data is based on a sequence of 32 vector quantizers: each of these quantizers applies not to single sample values, but to frames of 12 successive values. For any such frame, a scale factor is first computed and applied to the frame in order to map its values to $(-1, 1)$ so that a uniform midtread quantizer on $(-1, 1)$ may be applied.

The scale factor of a frame is determined – among a finite list of candidate scale factors – as the least value greater than or equal to all the absolute

values in the frame, or the greatest candidate scale factor value if no such value exists.

Define an increasing array of 64 scale factors with the following pattern:

```
>>> scale_factors
[... , 0.5, 0.62996, 0.79370, 1.0, 1.25992, 1.58740, 2.0]
```

2. **Bit Rate and Bit Pool Size.** Consider a sequence of 32 frames of 12 values, one by subband. Assume that the uniform quantizer used in the subband i can use b_i bits for the quantization any scaled sample, with $b_i \in \{0, 1, 2, \dots, 15\}$.

† Compute the total number of bits required to describe in every subband:

- the scale factor selected,
- the number b_i of bits allocated,
- the quantized values of the frame of 12 samples.

† At what frequency are produced batches of 32×12 sample values by the analysis filter bank ? What is the bit rate of the quantization, as a function of the total number of bits $b = b_0 + b_1 + \dots + b_{31}$ allocated for a frame of 12 (scaled) values ? We select a target bit rate of 192 kb/s. What is the value of the bit pool size b ?

3. **Bit Allocation Algorithm.** The bit allocation algorithm of the first psychoacoustics model of MPEG-1 is based on the comparison in each subbands of the noise level due to the quantization and the mask level generated from the psychoacoustic analysis. The algorithm that we implement iteratively allocates bits to make the noise-to-mask ratio approximately equal in each subband.

† Consider a single scale factor quantizer $[\cdot]$, applied to a frame of 12 values, that has computed the scale factor A and may use b bits by sample. Compute the quantization noise level

$$\text{noise level [dB]} = 10 \log_{10} \langle ([x] - x)^2 \rangle.$$

under the high resolution hypothesis.

Implement a function `allocate_bits` with arguments `frames` and `mask`, where `frames` is an array of shape (12, 32) and `mask` a sequence of 32 mask levels in dB, that returns the array `[b_0, b_1, ..., b_31]`.

The function shall implement the following algorithm:

while the bitpool is not empty:

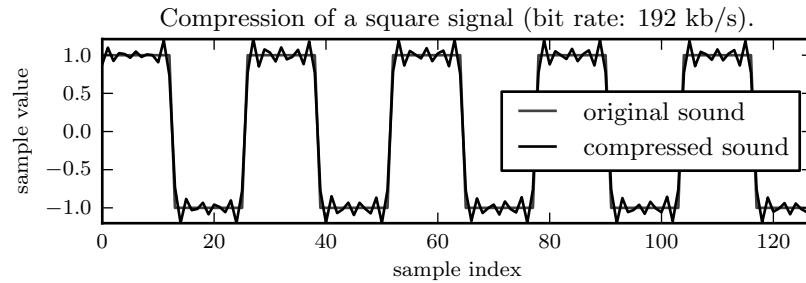
- i) compute the noise-to-mask ratio in dB for each subband

$$\text{NMR [dB]} = \text{noise level [dB]} - \text{mask level [dB]}$$

- ii) find the subband with the worst (biggest) noise-to-mask ratio,
- iii) allocate one extra bit from the bit pool to this subband, starting at 0.

Integration – Perceptual Compression

Implement a function `demo` that given a single-channel sound `data` will perform the compression and decompression by the methods of the previous section, display both signal and finally play both signals. Introduce an extra argument `bit_pool` that allows to compress with a bit rate different from the default.



† Use this function to test the compression of stationary signals (pure tones, square signals, white noise) and study in each case how much the bit pool size can be decreased without too much quality loss.