ELE 201, Spring 2014
Laboratory No. 5
MP3 Audio Compression

1 Introduction

In this lab, we'll explore compression rates that can be achieved with a fairly simplistic audio compression setup. Industry-standard algorithms such as MP3 (MPEG-2 Audio Layer III) or AAC (Advanced Audio Coding) are significantly more complex than the methods we will see here, even though the fundamental approach is the same.

Terminology: Assume that frequency is normalized to lie in $[0, 1]$ and we would like to analyze frequency bands $[0, 0.1], [0.1, 0.2], \ldots, [0.9, 1]$ separately. One approach would be to set up 10 band-pass filters corresponding to the frequency bands of interest, and filter our song through each of these. These 10 filters would be jointly referred to as the *filter bank* and each filter would be called a *tap filter*. Can you suggest benefits of analyzing frequency bands separately?

Our primitive audio compressor that will take as input an uncompressed `.wav` audio file and throw away redundancy that is perceptually irrelevant. The basic procedure is:

1. *Psychoacoustics*: Detect frequencies of interest in the audio and infer frequencies that are masked by others. Based on this, the psychoacoustic model returns a global 'mask' that guides us in compression;

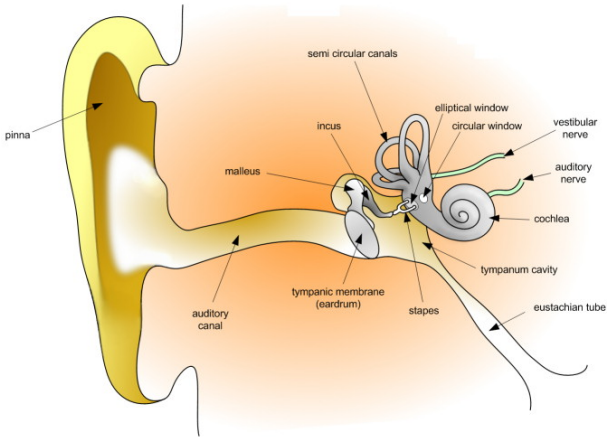
2. *Compression*: Run the audio through the *analysis* filter bank and quantize the output according to the mask computed above;

3. *Playback*: Run the quantized signal through the *synthesis* filter bank.

The lab is split into two parts. The first part focuses on the psychoacoustic model. The second part involves the signal processing and compression details.

2 Lab Procedure - Part 1

The purpose of this section is to understand better how the human aural system processes audio. As data compressors, we would like to understand the hardware (ear mechanism - eg. only frequencies in [20Hz, 20kHz] are detected) and software (post-processing done by brain - eg. spatial awareness through sound) used to handle auditory stimuli. The hope is that once we know which aspects of audio are emphasized, we can reduce the description complexity of the parts that are de-emphasized to achieve our goal of compression. For this reason, compression techniques such as MP3 are also referred to as *perceptual audio coders*.



2.1 Absolute Threshold of Hearing (ATH)/Threshold in Quiet

The ATH indicates the sound pressure level of a pure tone that is barely audible as a function of frequency. The curves you see in Figure 1 were produced from experimental data, so they can be thought to portray hearing thresholds of the 'average' human.

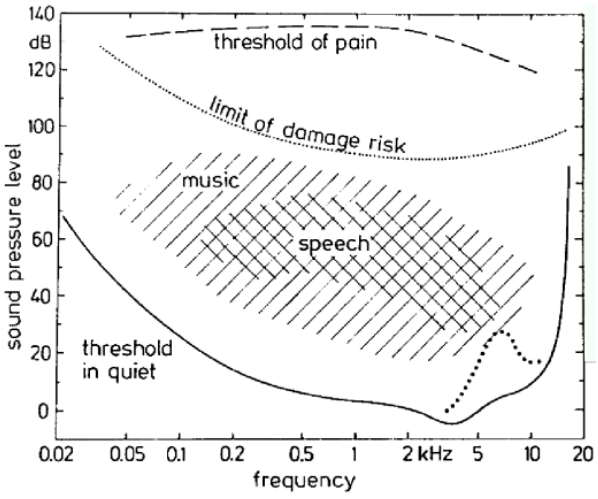


Figure 1: Thresholds for hearing.

As indicated, the y-axis shows sound pressure level (SPL) in dB (corresponding to loudness of the sound)

as a function of frequency in Hz. To be precise, the sound pressure level in dB is defined as

$$SPL(f) = 10 \log_{10} \left(\frac{p_f}{p_0} \right)^2,$$

where p_f is the physical pressures or deviation from atmospheric pressure induced by sounds at the frequency f and $p_0 = 20$ μPa, the hearing threshold for frequencies around 2kHz.

The auditory canal exerts a strong influence on the frequency response of our hearing. It acts like a half-closed pipe with a length of about 2 cm. Can you explain why this might imply the sensitivity of our hearing to frequencies around 4 kHz? Notice the dip of the auditory threshold around 4 kHz (the region of chalkboard squeaking). This high sensitivity is also the reason for high susceptibility to damage in the region around 4 kHz. Q1

It has been seen that exposure to high sound levels (such as loud music through earphones) produces temporary shifts in the ATH. After too many exposures, this temporary threshold shift results in a permanent shift (i.e. induced hearing loss). This shift is portrayed by the dotted bump above 3 kHz - 10 kHz.

2.2 Auditory Masking

The phenomenon of sounds being drowned out by louder ambient noises is termed *frequency masking*. Such masking is a direct consequence of the physical apparatus used in the ear to detect sound.

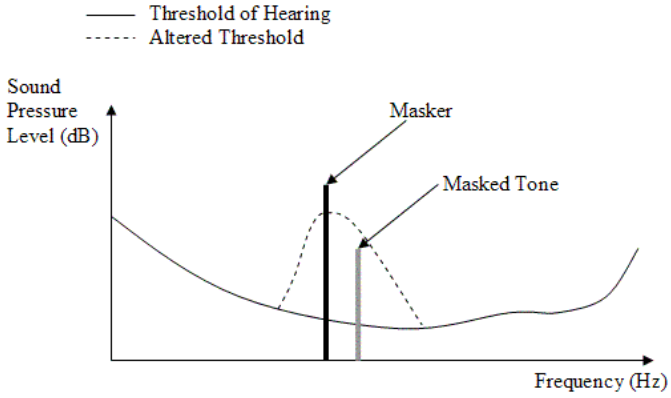


Figure 2: Frequency masking illustration.

Masking can also take place when the masker and maskee sounds are not played simultaneously. This is called *temporal masking*. For example, a loud vowel preceding a plosive consonant tends to mask the consonant. Such masking could also result from the temporary shift in ATH mentioned above.

Masking is one of the most important psychoacoustic effects used in the design of perceptual audio coders since it identifies signal components that are irrelevant to human perception. Experiments have been performed to deduce the shape of the masking threshold produced by sounds. The shape depends on both the amplitude and the frequency of the masking sound.

In measuring frequency masking curves, it was discovered that there is a narrow frequency range - the *critical bandwidth* - around the masker frequency where the masking threshold is flat rather than dropping off. This is especially apparent when you consider sounds that fall in frequency between two loud maskers. Figure 3 shows two examples of this. The first is narrow-band noise (not a pure tone) that is masked by two tones. The noise is centered at 2 kHz, and the masking tones are centered around this frequency as well with a volume of 50 dB SPL each. The plot shows the threshold at which the noise can be heard, plotted against the frequency separation of the masking tones. The second plot shows the same experiment but with two narrow-band noise signals masking a pure tone.

From the plots in Figure 3, can you conclude whether narrow-band noise or a pure tone is the more effective masker? Q2

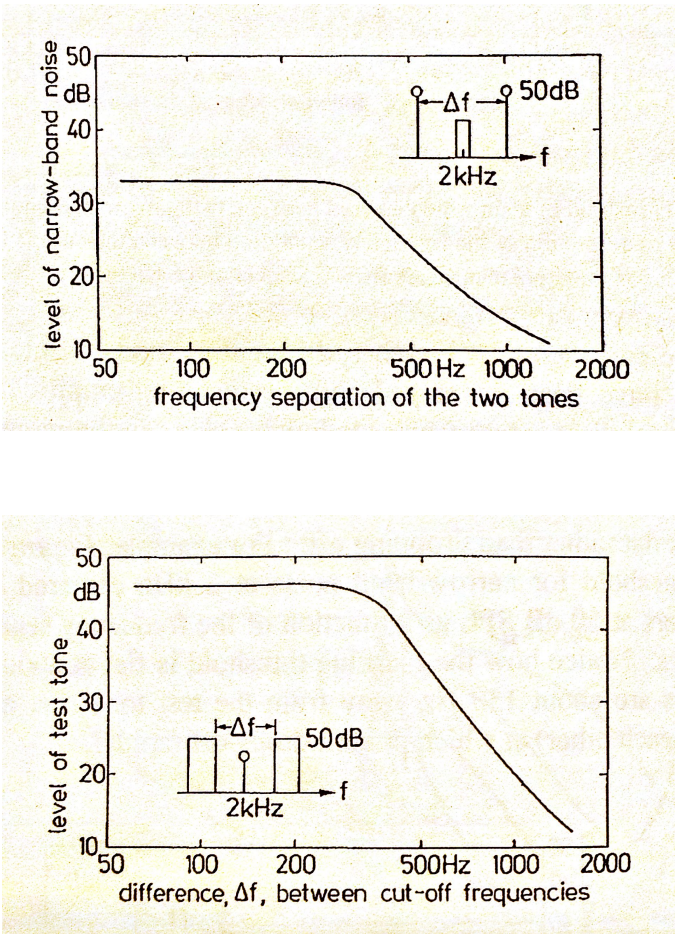


Figure 3: Inaudibility of one sound surrounded by two maskers.

By studying fluid motions in the inner ear, it has been deduced that the cochlea (the spiral-like object in figure of human ear) acts as a spectral analyzer! Sounds of a particular frequency lead to a displacement of fluid within the cochlea - essentially performing a 'frequency to space' transform. The displacement of the fluid conveys frequency information to the nerves lining the cochlea. It is believed that the critical bandwidths represent equal distances along the spiral.

This observation has led researchers to model the human auditory system as an array of band-pass filters, i.e. a filter bank with pass-bands of bandwidths equal to the measured critical bandwidths (which vary with frequency). In fact, a unit of frequency known as *Bark*, shown in Figure 5, was devised to match the change in frequency along the length of the spiral or, equivalently, the critical bandwidths that were observed experimentally. We shall use this scale for compression, as a perceptually relevant measure of frequency.

2.3 Implementation of Psychoacoustic Model

For the following, fix the length of the FFT to be 512. We provide the auxiliary functions:

- `ath(f)` (returns the threshold of hearing at frequency f),
- `dbinv(p)` (converts power in dB to power in magnitude),

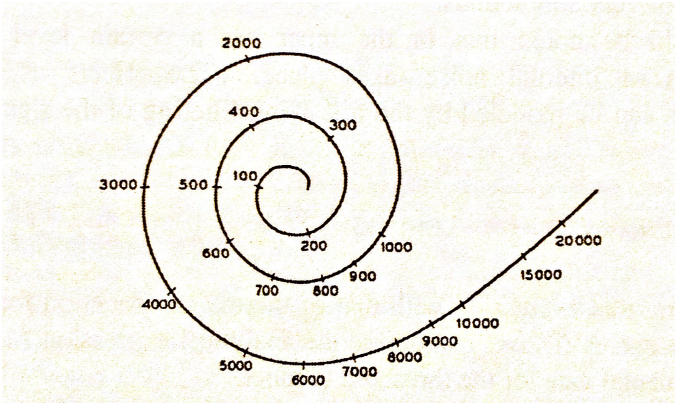


Figure 4: Human cochlea separates senses different frequencies at different locations.

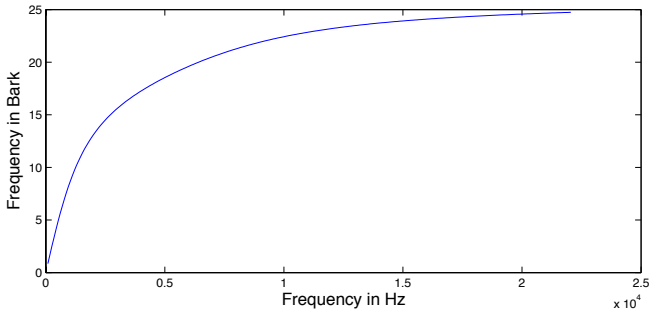


Figure 5: The *Bark* scale.

- `hz2bark(f)` (converts Hz to Bark),
- `psd(signal,fftlength)` (computes power spectrum of signal),
- `mask_threshold(type,location_of_masker_bin,psd_at_bin)`
  (returns the masking threshold for tonal/noise maskers; see `help mask_threshold`)

and an uncompressed audio clip `test.wav`. We also provide the functions `find_tones.m`, `noise_masker.m` and `findThreshold.m`, which are described below.

2.3.1 Tone Maskers

The function `find_tones.m` takes the power spectrum of audio as input and returns local maxima in the spectrum (this is similar to the procedure in Shazam). The local maxima (at frequency f) must be at least 7 dB greater than frequencies that are more than 1 index (~ 85 Hz) away within a frequency window of size $w(f)$, where

$$w(f) = \begin{cases} 350, & f \in [17, 5.5 \times 10^3] \\ 520, & f \in [5.5 \times 10^3, 11 \times 10^3] \\ 1030, & f \in [11 \times 10^3, 20 \times 10^3] \end{cases},$$

where all frequencies are in Hz. The window-lengths increasing with frequency is a consequence of the observed critical bandwidths increasing with frequency. Plot the tone maskers along with the signal PSD versus frequency (in both Hz and Bark). Use `stem` to plot the tones. M1

2.3.2 Noise Maskers

We provide a function called `noise_masker.m` which takes the power spectrum of audio and the tones found above as input and returns noise maskers in the spectrum. Only one noise masker is returned in every frequency interval $[j, j+1]$ for $j = 0, 1, \ldots, 23$ in Bark. The method is outlined below for completeness:

- For a frequency interval $[j, j+1]$, check if it contains any tones.
- If yes (tone at f Hz), then there can be no noise maskers within a frequency window of size $w(f)$.
- The remaining frequencies (if any) constitute the noise masker in $[j, j+1]$.
- The power of the noise masker is obtained by summing the powers of the above frequencies and then converting to dB.
- The frequency of the noise masker is the geometric mean of the above frequencies. Why the geometric mean instead of the arithmetic mean? Q3

Plot the noise maskers along with the signal PSD versus frequency (in both Hz and Bark). Use `stem` to plot the noise maskers. M2

2.3.3 Computation of Mask

Write a function called `global_threshold.m` that takes the tone maskers (from `find_tones.m`) and noise maskers (from `noise_masker.m`) as input and returns the global masking threshold of our psychoacoustic model. The provided function `mask_threshold` should be used to find the masks. We shall assume that the masks can be combined additively (do not add dB—use `dbinv`). Also, add on the magnitude of the ATH before converting to dB.

Plot the global masking threshold, the power of the signal, the ATH and all the tonal/noise maskers that were found on the same plot, with frequency in Bark. We provide the function `findThreshold.m`, which assembles the above routines into a single one that takes as input a signal and returns its global masking threshold. M3

3 Lab Extras

If you have time, here are some extra activities to try. This section is not to be turned in for credit but is highly recommended.

3.1 Absolution Threshold of Hearing Experiment

Preferably using earphones, set the volume to an adequately low level and play sinusoids of varying frequencies using Matlab. Can you reproduce the trend shown in Figure 1? In particular, do you notice a sharp dip around 4 kHz? You may keep the volume fixed across frequencies and expect that a frequency with lower ATH will sound louder than one with a higher threshold.

3.2 Masking Experiment

Can you exhibit masking of one tone by another close in frequency? Add two tones together, with one much quieter than the other, and see if you can make the quite one vanish even though it would still be audible if played by itself.

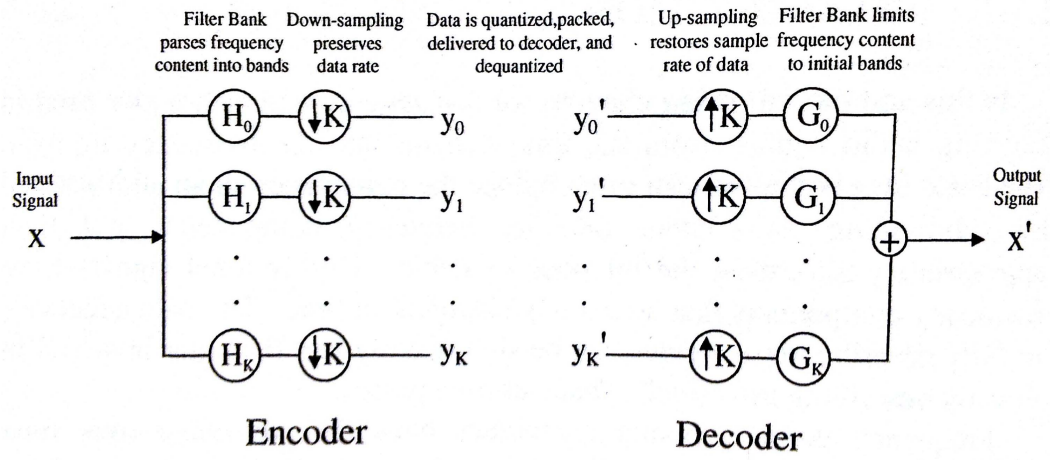
4 Lab Procedure - Part 2

In the previous part, you worked on a psychoacoustic model that generates a masking threshold for a given sound clip. Next, we shall perform audio compression by using the threshold to estimate the number of bits to be assigned in our quantization scheme. For this lab, we provide *all* the code required for the implementation of filter banks, quantization and measurement of compression rate.

4.1 Polyphase Filter Bank

The motivation behind using filter banks before quantization is that we can often reduce redundancy in an audio signal by subdividing its content into frequency components and then appropriately allocating the available bits. Highly tonal signals have frequency components that are slowly changing in time. It can be significantly simpler to efficiently describe these signals than to try to directly describe their shape as a function of time. Furthermore, this ability to allocate resources (bits) differently across the spectrum allows us to both reduce the overall quantization noise level and take advantage of the psychoacoustic model.

The basic idea is to pass the signal through a bank of filters (analysis) that parse the signal into K bands of frequency. The signal from each band is then quantized with a limited number of bits, placing most of the quantization noise in bands where it is least audible (or is masked well by signals of interest). The quantized signal is then sent to a decoder (synthesis) where the coded signal in each band is dequantized and the bands are combined to restore the full frequency content of the signal. An issue with this approach is that we have multiplied our data rate by a factor of K. To avoid raising the data rate, we down-sample by a factor of K. Remarkably, it is possible to design filter banks such that the original signal is fully recoverable from the down-sampled data.



In a practical filter bank system, the above filters will not be ideal, and decimation of the filter outputs will result in aliasing errors. In terms of the z-transform, a generalization of the DTFT for any complex-valued z (replace the complex exponential in DTFT definition with z), we will have the z-transform of the reconstruction given by

$$\hat{X}(z) = T(z)X(z) + \text{terms due to aliasing.}$$

If we can design the analysis and synthesis filters such that there are no aliasing terms and $T(z) = c \cdot z^{-k}$, where c is a constant and $k \in \mathbb{N}$, then the system is said to have the *perfect reconstruction* (PR) property. This will ensure that the output of the system is the original signal delayed by k time-steps, up to rescaling (by c). We provide a PR filter bank for this lab, but we shall not discuss its construction here.

In the provided file `lab5_p2.m`, we have included a routine to generate `h` and `g`, analysis and synthesis filter banks respectively, that constitute a PR system. There are 32 filters and the length of each one is 64, so that `h` is a 32×64 matrix. Verify that these filters partition the frequency spectrum - use the provided

vector of frequencies `f1`. Note that the sum of the powers is flat across frequency. Explain how `h` relates to `w`. M4

Puzzle: Have you ever converted a `.mp3` file to a `.wav` file? What you should observe is that the file size increases - sometimes by a factor as large as 10. We know that the `.mp3` file was compressed audio, while `.wav` is the format of choice for uncompressed audio. Why do we observe an increase in size - surely, there is no way to de-compress compressed audio? The `.wav` file cannot contain more interesting information than the `.mp3` file did. Is `.wav` just an inherently inefficient format? Q4 Q5

4.2 Quantization

In the same file `lab5_p2.m`, we also provide code to create `xQ`, a quantized version of `x` - using `b` bits ($2 \leq b \leq 16$, since the original data rate is 16 bits/sample). From lecture, you might remember that the finer our quantization, the weaker will be the correlation between the error and the signal. The error (`x-xQ`) represents the amount of information lost. Ideally, we would like this error to resemble noise, so that no relevant information is lost. Does the error sound like noise? What happens as you change the number of bits used? What is the minimum number of bits for which the error sounds like noise? Is this behavior confirmed by the power spectrum of the noise? D1 M5

4.3 Implementation

The file `encode.m` takes as input a sound clip, its frame rate and bits used per value (this is returned by `wavread`). It returns the compressed audio vector and plots the FFT spectra of both versions, along with the spectral error. It also displays the compression rate in the console.

To process the input sound clip, it breaks the clip into overlapping chunks of 11.6 ms (this windowing needs to be done carefully in order to avoid introducing artifacts into the song). For each chunk, it calls your psychoacoustic model to retrieve the masking threshold that will be used for compression. We have provided a simple bit assignment scheme in the file `performEncoding.m`.

What is the compression rate you see for the provided file `test.wav`? Compare the spectral error you see with the spectral error from quantization alone. What happens when you use the ATH instead of the masking threshold (in `encode.m`, replace the call to `findThreshold`)? Can you use an alternative bit assignment (in `performEncoding.m`) to get a better compression rate? For instance, you could use your knowledge of the filter bank design or tonal information about the signal from your psychoacoustic model. Does your bit assignment scheme work well for compressing other audio files? D2 D3 D4

{What do you observe when `w` (defined in `encode.m`) is the all 1's vector? Do the filters still partition frequencies with sum of powers constant? Explain.} Compare the bit rates of the compressed file to the original bit rate. How does the bit rate compare to typical `.mp3` files? Can you suggest additional techniques or modifications that would allow us to get higher rates of compression? Alternatively, using your knowledge of sampling from class, can you come up with simple compression schemes that achieve similar compression rates with comparable quality? Q6 D5 D6

5 References

- *Introduction to Digital Audio Coding and Standards* by M. Bosi and R. E. Goldberg
- *Psychoacoustics: Facts and Models* by H. Fastl and E. Zwickler
- *Watermarking Systems Engineering: Enabling Digital Assets Security and Other Applications* by M. Barni and F. Bartolini
- *WAVS Compression* by A. Chen, N. Shehad, A. Virani and E. Welsh