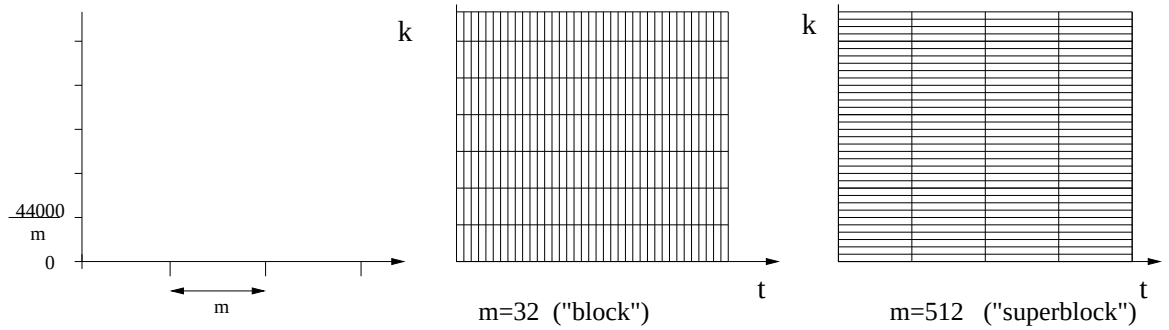
MP3-like lossy audio encoder/decoder

Suppose we have an audio signal $X(t)$. First let's consider a spectrogram that is defined by the blocks of size $m = 32$. Frequencies of the DCT are $k = 0, 1, \ldots, 31$. Each block is $\frac{32}{44{,}000}$ seconds, which is a total of about $\frac{3}{4}$ ms. That is, we have 32 numbers (the $Y_j(k)$ coefficients) every $\frac{3}{4}$ ms. The k values represent temporal frequencies $\frac{k}{2}$ cycles per $\frac{32}{44{,}000}$ seconds, so that $k = 1$ represents about 700 cycles per second, $k = 2$ is about 1400 cycles per second, etc.

In order to encode the $Y_j(k)$ values in each block j (namely $k = 0, \ldots, 31$) using lossy compression, we need to choose a quantizer width Δ. On the one hand, we would like to take account of sensitivity and masking effects that we discussed last class, and choose the Δ value based on these effects. In particular, since the masking effects vary over time and over frequency, depending on the audio signal $X(t)$, it would be great if we could allow the Δ values to vary with block j and with frequency k. On the other hand, we cannot use a different Δ value for each (j, k), as we would need as many Δ values as we have samples in the original signal, so we would have little hope of achieving compression!

A second consideration is that the k values (32 of them) for each block only crudely represent the various sound frequencies, relative to the resolution we saw earlier for much larger blocks. The way to think about this is that if we have only 32 samples which are over a very short time interval, then these samples cannot distinguish the presence of two sound frequencies that are very similar e.g. 500 Hz vs. 550 Hz. But we need higher resolution of the frequencies than that in order to estimate masking effects in the sound, since masking occurs when multiple *nearby* sound frequencies are different from zero. The crude resolution of about 700 Hz that we get with blocks $m = 32$ is much too coarse to analyze masking.

The solution to these two problems is as follows. First, we analyze masking effects by considering a second spectrogram whose blocks are of size $m = 512$. We call these *superblocks*. (There are $16 = \frac{512}{32}$ blocks per superblock.) As discussed in lecture 31, there are 512 values of k for the superblocks and these have a resolution of about 43 Hz. This is much smaller than the 700 Hz resolution for the (sub)blocks. It turns out that 43 Hz resolution is sufficient for estimating masking.



The main idea of MP3-like methods is to use the superblocks ($m = 512$) to analyze the masking effects and to determine a set of Δ values that vary slowly across frequency and across time. Masking effects themselves vary slowly across space and frequency, as indicated last class: the amount by which one sound frequency k_M can mask another k_T depends on the distance between them $|k_M - k_T|$ and distance is a continuous function (obviously). Similary, if two sounds occur at nearby times, then the amount of masking depends continously on the temporal interval between them.

So, specifically, what do we do ? Rather than having a different Δ value for each frequency $k = 0, \ldots, 31$ and each subblock $j = 0, ..\frac{n}{m}$, where n is the number of samples in the audio signal, we use a different Δ_k value for each $k = 0, \ldots, 31$ but we use the same Δ_k value for each of the $\frac{512}{32} = 16$ blocks in a superblock. Thus, rather than have 512 different Δ values for each superblock (each 16 blocks), we will have 32 values per superblock. Encoding these Δ_k values for each superblock costs extra bits of course, but since there are far fewer Δ_k values than samples in the superblock (i.e. $32 \ll 512$), one can potentially still achieve compression.

Encoder

Given Δ_k for a superblock, you might expect that the quantization of the $Y_j(k)$ values within each (sub)block j are done similarly to what we discuss with image coding, namely use a mid-tread quantizer to compute *levels*:

$$l_j(k) = round(\frac{Y_j(k)}{\Delta_k})$$

to choose the level which is encoded. MP3 doesn't quite do this, however. The main reason is that it is difficult to say in advance what are are the probabilities of the various levels $l_j(k)$ that one will find. If the audio signal has a large value of $Y_j(k)$ for some (j, k) and the frequency components driving this $Y_j(k)$ are relatively isolated so that there is very little masking, then the levels $l_j(k)$ over the 16 (sub)blocks can each be *quite* different from zero. On the other hand, if there is masking present, then the Δ_k values chosen will be large, then these 16 $Y_j(k)$ values will tend to be relatively small.

To compress well, we need a *code* for the levels $l_j(k)$ whose codeword length is appropriate for the probabilities of these levels. If the encoder only tells the decoder what are the Δ_k values for each block, then this is not sufficient for the knowing the range of $l_j(k)$ values. MP3 deals with this by also telling the decoder what is the range of the $l_j(k)$ values for each superblock ($m = 512$) and for each k. The idea is simple: in addition to coding the Δ_k values for each superblock, the encoder also encodes

$$l_{max}(k) \equiv \max\{ \mid l_j(k) \mid \; : j = 0, \ldots, 15 \; \}.$$

This tells the decoder that the range of values of the $l_j(k)$ for that superblock is $-l_{max}, \ldots, l_{max}$. The encoder and decoder agree in advance on what code to use for each possible l_{max} value. For example, a fixed length code with $\lceil \log_2 l_{max} + 1 \rceil$ bits could be used.

To summarize, for each superblock of 512 samples, the encoder sends

- 32 Δ_k values
- 32 $l_{max}(k)$ values
- 512 levels $l_j(k)$, which are encoded in a way that depends on the $l_{max}(k)$ values.

You may be skeptical that compression can be achieved by this. Each sample in the audio file is 2 bytes. Thus, we can only afford to spend a few bits on each of the Δ_k, l_{max}, and $l_j(k)$ values, on average. Amazingly, this is good enough. One can achieve compression ratios of about 10:1 with hardly any noticable loss.

Decoder

The decoder (the MP3 player) is much simpler than the encoder. The decoder does not have to do any masking calculations. For each block, the decoder is given the values mentioned above. From the Δ_k and levels $l_j(k)$ for each superblock, namely for each j compute the 32 $\Delta_k l_j(k)$ values (over all k) and take the inverse DCT to get the X values for that block.