Creative Computing II

Audio Signals and Fourier Analysis

1st December 2010

This lab sheet involves visualising and decomposing signals into their sinusoidal components using *Octave*, and then using that decomposition to generate sounds using *Processing*.

1. This part of the lab involves reproducing numerically the result given in the lecture about the sinusoidal components of a square wave.

   (a) In *Octave*, construct a vector representing one second of a square wave signal with frequency 1Hz and amplitude 1, sampled at 44.1kHz.

   ```
   x = [ones(1,22050) -ones(1,22050)];
   ```

   (b) To find the first coefficient b_1 of the Fourier decomposition of your square wave, calculate twice the mean value of the Hadamard product of your signal from part 1a with a sine signal, with amplitude 1, frequency 1Hz, sampled at 44.1kHz. Note down your answer.

   Using the expression `mean(x.*sin(2*pi*1*[0:44099]/44100))`*, we get an answer of about 1.27.*

   (c) Repeat part 1b, but with sine signals of frequency 2Hz, 3Hz, 4Hz, 5Hz, ..., noting down the answer each time. Do you notice a pattern?

   You should notice that the even frequencies give answers of near enough zero (remember the scientific notation `e-17` *meaning* $\times 10^{-17}$*); the odd frequencies should be related to the answer from part 1b by the reciprocal of the frequency ratio.*

   (d) Repeat part 1b, but with cosine signals of frequency 1Hz, 2Hz, 3Hz, ...; do you detect a pattern?

   All the values should be close to zero, because the square wave as we've defined it is an 'odd' function (for odd functions f, $f(-x) = -f(x)$) whereas the cosine function is 'even' ($f(-x) = f(x)$), so their product is odd, and vanishes on average.

   (e) Using the patterns you have observed, make a guess for the complete decomposition of the square wave signal into its sinusoidal components. Construct a signal using the first 5 terms of the expansion, and use *Octave* to visualise it. Does it look like a square wave?

   This should lead you to the expression presented in the lectures, $s(x) = \sum_k \frac{1}{2k+1} sin(2\pi(2k+1)t)$*. Visualising this to 5 terms should be enough to convince you that it's plausible, at least; if you visualise multiple terms, you might observe an interesting phenomenon (called the Gibbs phenomenon) near the edges of the square wave.*

2. This part of the lab involves using Minim and *Processing* to synthesise the sound of a square wave.

   (a) Using the example from the lectures as a starting point, write a sketch to generate a square wave sound at a given frequency using Minim and the `AudioSignal` class. Run your sketch with the frequency of the wave set to 440Hz; listen to the output, and try to describe it.

See the 'minim signal' sketch on the course website for the technical answer to this part and the next. I would describe the sound of this (exact) square wave as fairly rough and unpleasant.

(b) Adapt your sketch from part 2a to use, instead of an exact square wave, an approximation to it using the first five terms of the sinusoidal decomposition. Again, listen to the sound output, and describe it in words; also, describe any difference you can between the exact and approximate square waves.

In contrast to the exact square wave, this approximation sounds fairly pleasant and harmonic.

(c) Add more sinusoidal terms to the signal generation in your sketch. Does the output sound change noticeably at any point? If so, try to explain why.

It is possible that, at some point, the sound output of the additively-synthesized square wave becomes choppy and clicky; this is most likely a symptom of having too much work to do in one buffering cycle. A buffer of 2048 entries lasts for about one twentieth of a second; if the computations being done to fill that buffer (and all other activities performed by the program, such as drawing a waveform) take longer than this time, then the buffer will be shipped to the audio system before it has been correctly filled, leading to audio artifacts. It is important in real-time synthesis to work efficiently, and there is significant research in the construction of rapid evaluation algorithms for synthesis.

3. Repeat the above process (decomposition and synthesis) for another simple wave form, such as a triangular or sawtooth wave form.

Other resources:

- Roads, C. *The computer music tutorial*, Chapters 4 (Additive Synthesis) and 13 (Spectrum Analysis). MIT Press (1996)