Assignment 8: Synthesis Beyond Two Sine Waves

CS 4347: Sound and Music Computing

due Wednesday 01 April 2015, 11:59 pm

NOTE: For inquiries on this assignment, please contact FANG Jiakun (a0123777@nus.edu.sg).

0. Each section (numbers 1–2) should have its own source code file.

1. Use additive synthesis to construct a band-limited sawtooth wave at $f = 1000$ Hz lasting 1.0 seconds with a maximum amplitude of 0.5, using $F_s = 44100$.

   The formulae for sawtooth wave is:

$$x_{sawtooth}(t) = -\frac{2A}{\pi}\sum_{k=1}^{M}\frac{1}{k}\sin(k2\pi\frac{f}{F_s}t)$$

   M is the maximum possible number of sine waves without aliasing.

   Submit:

   - a png showing the time-domain perfect sawtooth wave and your reconstructed one. Only include 5-6 cycles in this figure. The title of your plot should state how many sine waves you used, which should be the maximum possible without aliasing. You may use `scipy.signal.sawtooth()` to create the perfect sawtooth wave.
   - a png showing the dB-magnitude FFT (not a spectrogram!) of a perfect sawtooth wave and your reconstructed one. Use FFT length of 8192 (so ignore the remaining $44100 - 8192 = 35908$ samples).
   - write 1 paragraph contrasting what you hear when you listen to a perfect sawtooth wave and your reconstructed one. You do not need to submit the `wav` files, but you must submit the paragraph.
   - your python source code.

2. Generate sine waves at different frequencies using a look-up table.

   - Create 1 look-up table with 16384 samples. This must contain a single cycle.
   - Use that look-up table to create sine waves at $f = 100.0$ Hz and $f = 1234.56$ Hz. Use $F_s = 44100$, and generate 1.0 seconds of audio. (hint: before proceeding, plot the look-up table to ensure that it only contains 1 cycle of a sine wave. 1 cycle means that the final value in this array should *not* be 0; it should be slightly below 0)
     For each sine wave, create 3 versions:
     - no interpolation (using the look-up table)
     - linear interpolation (using the look-up table)
     - perfect version (using `numpy.sin()` directly)
     - Calculate the maximum error of the look-up table versions: given `LUT_sine_wave` and `perfect_sine_wave`, do:

```
max_error = numpy.max(numpy.abs(LUT_sine_wave - perfect_sine_wave))
max_audio_file_error = 32767*max_error
```

- Repeat the above using a look-up table with 2048 samples.
  The above `max_audio_file_error` is not a completely accurate representation of potential errors in the `wav` files, but it is close enough. If you use `round()` for the "no interpolation", the error should be approximately 2π in the worst case for 16384 samples, and approximately 100 in the worst case for 2048 samples. If you did not use `round()`, then these errors will likely be twice as big.
  With interpolation, the best result should have an error less than 0.001.
- Submit:
  - a text file giving the `max_audio_file_error` of the 2 sine waves using no interpolation and linear interpolation for 16384 and 2048 samples. Your text file should be formatted as follows:

```
Frequency  Interpolation    16384-sample    2048-sample
100Hz      No               err_1           err_2
           Linear           err_2           err_4
1234.56Hz  No               err_3           err_6
           Linear           err_4           err_8
```

  - your python source code.

Grading scheme:
- **3/6 marks**: files for 1. additive synthesis of a band-limited sawtooth
- **3/6 marks**: files for 2. the look-up table