CSc 637 Spring 2009
Project 2: Envelope approximations in analysis/resynthesis

due in class (by 6:15), Monday 4/6/2009

Preliminaries: spectrum analysis data files

For this project, you will work with some files containing spectrum analysis data for three instrument tones: flute, piano, and trumpet. The soundfiles (*.snd) and analysis files (*.pv.an) are on thecity.sfsu.edu at

~hsu/637/P2/Data

You won't need to generate the analysis files from the wav files. However, if you are curious on how this is done, you can download the *sndan* sound analysis package. See:

`http://ems.music.uiuc.edu/beaucham/software/sndan/`

Envelope approximation

The application you write will work with data from the .an files. The `cpan` utility is provided on thecity at

~hsu/637/P2/Code

It extracts amplitude or frequency data from a .an file, and prints the information in ASCII. Suppose you have an analysis file Piano.pv.an in your current directory. Here's a sample run to generate amplitude data:

```
thecity% ./cpan Piano.pv.an

Read file ./Piano.pv.an
Date Recorded:
Instrument: , Performer:
Comments:
test piano
Pitch is , dynamic is
Base analysis frequency = 311.00 Hz
Time between frames = 0.001608 sec
Original sample rate = 22050.000000
Duration of sound is 2.494
Magfr format is type simple
Number of analysis frames is 1551
Number of harmonics per frame = 35
Use nhar = 35
# harmonics = 35 # timepoints = 1551
Amplitude data:
harmonic       1        2        3        4        5
```

```
frame 0 0.000000 10925.22  2503.12  2923.30  1618.10  1891.79
frame 1 0.001608 11026.23  3191.47  3394.77  2131.38  2613.73
frame 2 0.003215 10094.43  2686.02  3501.59  2189.07  3009.87
frame 3 0.004823  9115.93  2586.31  3673.34  2081.71  3167.78
frame 4 0.006431  8455.43  2658.59  3808.13  2019.40  3302.75
frame 5 0.008039  8540.69  3210.84  4036.73  2217.07  3407.60
frame 6 0.009646  9689.18  3805.80  3917.17  2470.70  3334.16
frame 7 0.011254 10375.40  3394.01  3903.99  2533.21  3365.91
frame 8 0.012862 10178.14  2996.96  3929.06  2509.46  3392.45
frame 9 0.014469 10068.75  3275.64  4011.97  2437.77  3349.84

Frequency deviation data:
harmonic        1       2       3       4       5

frame 0    10.95   -14.09    -7.58   -10.45     9.63
frame 1     8.77     4.33     6.89    -6.13     7.68
frame 2     2.94    31.11     2.79    -0.87     7.60
frame 3     4.24    26.09     3.41     6.99     9.45
frame 4    -0.88    27.23     7.12    13.79     9.81
frame 5    -6.33    -3.97     2.90     4.49     8.63
frame 6     0.99    -7.67     3.15     6.07     7.66
frame 7     4.56    14.15     1.50     2.26     7.17
frame 8    -3.63    13.48     0.80     1.84     7.15
frame 9    -8.29    -0.33     0.48     4.22     8.06


etc etc etc
```

The data starts at time = 0 sec. At 0 seconds, the amplitude of harmonic 1 is 10925, harmonic 2 is 2503 etc etc. At 0.0016 seconds, the amplitude of harmonic 1 is 11026, harmonic 2 is 3191 etc.

The source code for cpan is provided, to illustrate how to extract data from the .an files.

Project specification

In this project, you will write an application that takes as input a file [inst].pv.an in the format provided (generated by sndan), and a number of breakpoints per second N_{bk}, not counting the first and last breakpoints. (Hence, for a 2-sec sound, with $N_{bk} = 3$, the total number of breakpoints is 3 * 2 + 2 = 7.) Your application will perform approximations for the amplitude and frequency envelopes for each harmonic of the tone, based on the procedure discussed in class. Each amplitude envelope becomes approximated by line segments with $2+N_{bk}$ breakpoints per second (over the duration of the tone). Finally, your application generates an SAOL and a SASL file in the MP4-SA format, specifying the new approximated envelopes that you calculated. When compiled, the SAOL and SASL files will synthesize a tone that is similar to the original acoustic instrument tone, but with greatly reduced data requirements.

If you would like to create your own phase vocoder analysis files (which is not necessary), try the *gopvan* script, a friendly interface for the *pvan* phase vocoder

software. gopvan works with most common file formats, or you can use *Audacity* to convert your soundfile to a supported format.

You should test your program on the three instrument tones and make sure they work properly. For each tone, process the first 30 harmonics only; discard the rest of the harmonics. Also, try N_{bk} = 1, 3 and 8 breakpoints per second. Expect the synthetic tone to sound bad for 1 breakpoint per second, and get progressively better as you increase the number of breakpoints.

Command line format:

Your submission should be a single tar file that expands into a directory named [your last name].P2 (for example, Hsu.P2). The directory should contain a usable Makefile for Unix systems. The user should be able to compile all binaries by typing at the Unix command line

```
make specCompress
```

To run the application, the user types at the Unix command line

```
specCompress [.pv.an file] Nbk [.saol file] [.sasl file]
```

A .saol and .sasl file will be generated. When the two files are compiled using sfront, a soundfile is produced which contains the original sound after line-segment envelope approximation and resynthesis.

Short report:

Write a short (1-2 page, double-spaced, 12-point type) report. You should summarize how effective the approximation procedure was, giving examples of problem envelopes that were difficult to approximate properly (if any). For each instrument tone, indicate how good the approximations were, for 1, 3 and 5 breakpoints per second, and the smallest number of breakpoints that are sufficient for a reasonably sounding approximation of each tone. *[Spring 2009 version: you don't have to write the short report]*

Submission:

Submit a paper copy of your report and your code, and (by email) a copy of the source code of your application. All instructions for compiling and running your application should be in the program header. Post your best automatically resynthesized soundfiles to the iLearn forum before class on the due date.