Music 420
Winter 2005-2006
Homework #4
Interpolation, Doppler, Leslie, Flanging
145 points
Due in one week (Feb/16/2006)

Theory Problems

1. (15 pts) As discussed in the text (and lecture overheads), the transfer function of first-order allpass delay-line interpolation is is

$$H(z) = \frac{\eta + z^{-1}}{1 + \eta z^{-1}}.$$

Find the *phase delay* $\Delta(\omega)$, and its dc limit $\Delta(0)$.

2. (10 pts) What range of η corresponds to $\Delta \in [0.5, 1.5]$?

3. (20 pts) Find an interval for η such that the maximum impulse-response decay-time is $t_{60} = 10$ ms. Assume a sampling rate of $f_s = 10$ kHz.

Lab Assignments

Lab assignments are due one day after the theory portion of the homework (typically Friday afternoon at 5pm). Lab assignments are to be submitted electronically at http://coursework.stanford.edu. For each assignment, submit a single archive (`zip` or `tar.gz`) file containing all code/figures/output for the assignment. For each problem in the assignment, create a sub-directory in the archive named `hwX_pY`, where `X` is the homework number, and `Y` is the problem number, and place all code/figures/output for that problem in that sub-directory.

Starter code and soundfiles for this assignment are archived in `hw4stuff.tar.gz`.[1] To unpack, first save it into your STK `myproj` directory (typically `~/stk/stk/myproj`). Then use the following command in a terminal at the same directory level: `tar -zxf hw4stuff.tar.gz`.

1. (50 pts) **Doppler**

Write an STK class called `Doppler.cpp` which simulates the *Doppler effect* using an interpolating time-varying delay line. The inputs to the `tick:` method should be:

[1] http://ccrma.stanford.edu/~jos/hw420/hw4/hw4stuff.tar.gz

- the input signal sample $x(n)$
- a Doppler-shift parameter $\tilde{v}(n) = v_s(n)/c$, which stands for the current speed of the source toward the listener, $v_s(n)$, normalized by sound speed, c.

The formula for the Doppler frequency shift is

$$f(n) = \frac{f_0}{1 - \tilde{v}(n)}$$

where f_0 is the frequency emitted by the source at rest.

Write a test program which varies the Doppler parameter sinusoidally as

$$\tilde{v}(n) = 0.1 * \sin(2\pi t)$$

and run the Doppler shifter on a test sinusoid given by

$$x(n) = \sin(2\pi 220 nT), \quad 0 \leq nT \leq 1.$$

where $T = 1/10000$ denotes the sampling period. Listen to the result to make sure it has no artifacts and that the pitch goes up to a maximum, down to a minimum, and back up to the original pitch.

Change `DelayL` to `DelayA` and explain in what ways the performance differs from the previous case.

Turn in your commented program listings and a printout of the first 100 numbers (or so) of your simulation.

2. (30 pts) **Leslie**
Look at the STK class `Leslie/Leslie.cpp` given as starter code. It consists of four `Doppler` instances you have just created, each delayed by some amount, and then put through a reverberator. For a general description of the Leslie effect, see `http://www.theatreorgans.com/hammond/faq/mystery/mystery.html` and/or `http://www.geofex.com/Article_Folders/lera/lera.htm`.

   The basic design here is that the direct signal from the main rotating horn gives rise to one of the Doppler units, and the other three correspond to the first three side cabinet reflections. In principle, the reverberator represents the remaining cabinet reflections, plus any desired room reverberation.

   Control parameters you can experiment with are the speed and the length of the rotating horn along with the dimension of the cabinet. Optional parameters you may include are the relative phase of the Doppler modulations (e.g., make them uniformly distributed from 0 to 2π as a default) and any reverberation parameters you want to bring out.

   (a) Complete the code in `Leslie.cpp` in the `tick` method.

(b) Write a test program that takes any sound file and runs it through the Leslie simulator, varying the speed smoothly from 1 Hz to 10 Hz and back down to 2 Hz over the duration of the sound file.

(c) Draw a diagram of the `Leslie` according to the parameter names and settings in the code, showing the cabinet dimension, horn length and its position at a time instant etc. Also, describe how the multi-path delays are approximated in the simulator.

Submit your commented program listing.

3. (20 pts) **Flanging**
Look at the STK class `Flanging.cpp` given as starter code. For more information specific to this assignment, see

`http://ccrma.stanford.edu/~jos/pasp/Flanging.html`.

(a) Complete the `tick` method in `Flanging.cpp` and set up the parameter values in the constructors to give a nice effect when testing in the main program.

(b) Write a test program that takes as input (i) noise generated from an STK class called `Noise` (ii) any sound file, which then goes through the flanging effect.

(c) Test your flanger on the file `alb.wav` (also available in Ogg Vorbis format as `alb.ogg`).

Submit your commented program listing. Make sure it shows both cases, perhaps with one commented out, of noise input and sound file input.