E85.2607 – Advanced Digital Signal Theory
Assignment 4: Analysis-Synthesis effects
Due: 2010-04-25 by 11:59pm

Instructions

Send me an email containing your solutions by the due date listed above. Please attach a zip file containing your solutions, including the following:

1. A writeup including any requested plots as a **PDF** file. All Word documents will be ignored.
2. Any Matlab code you wrote as separate plain text files.
3. Any interesting sounds you generated.

Your submission should be your own work. If you're having trouble please send me an email or come to my office hours.

Lateness policy

Please note the due date at the top of this page. Late submissions will be penalized by 10% per day, unless an extension has been granted. *This means that if you submit your homework at 12:01am, you will lose points.*

Get started early. Extensions will not be granted on the day the assignment is due. If you think that you will need extra time ask for an extension in advance.

1 Adaptive Filtering with LPC

This problem has been blatantly plagiarized from Matt Hoffman

In class we discussed how and why Linear Predictive Coding works to give us an all-pole filter that is optimized to match the rough spectral shape of an input window of samples. Actually implementing LPC can be slightly hairy, but fortunately it's already implemented for us in MATLAB's signal processing toolbox.

Write a MATLAB function that takes in two signals, and filters one of them (the "input" signal) using a all-pole filters obtained by applying LPC to short windows of the other (the "control" signal). The goal is to impose the time-varying spectral shape of the control signal on the input signal, much like a channel vocoder would. Your function declaration should look like this:

```
function y = lpccvoc(ctrl, input, order, winsize)
% y = lpccvoc(ctrl, input, order, winsize)
%
% ctrl is the control signal, input is the input signal, order is
% the order of the all-pole filters we want to use, and winsize is
% the size in samples of the analysis windows we'll be using. The
% output should be stored in y. You can assume that ctrl and input
% are the same length.
```

Please make sure that your function conforms to this declaration.

The basic steps your function should take are:

- For every winsize samples, use MATLAB's lpc function to calculate a set of order filter coefficients from the appropriate winsize samples of the ctrl signal.

- Apply an IIR filter with the coefficients returned by lpc to the appropriate winsize samples of the input signal. Zero pad the input window by an additional winsize samples so that the filter has a chance to decay completely.

- Multiply the output window obtained in the previous step by the RMS power of the window of samples from ctrl.

- Add the result of the previous step to the output signal at the appropriate time offset, overlapping the zeropadded output signals.

The zero-padding is a bit of a pain, but failing to do it can lead to very undesirable audible artifacts resulting from cutting off the decay of the all-pole filter every winsize samples.

A good way to test your function is to apply the spectral envelope of a few seconds of some music or speech to an impulse train. As you use higher and higher order filters, you should get more and more detail from the control signal in your output.

Find some interesting sounds to run through your function (and submit the results). What happens if the input signal is white noise as opposed to an impulse train?

2 Analysis-synthesis effects: Freddy's Revenge

In this problem you're going to explore a variety of effects built on the analysis-synthesis techniques we've discussed in class. Dan Ellis has released Matlab code for many of these algorithms on the web, so you're not going to have to implement them yourself.

Process the first 5 seconds of
`http://www.ee.columbia.edu/~ronw/adst/homeworks/hw2-vocals.wav`
using each of the effects listed below. You can find some other fun sounds to play with here, here, or here. For each effect you should include:

1. The resulting sound file. Please use the filenames listed in parenthesis.
2. A spectrogram demonstrating the effect.
3. The Matlab code used to create the effect. This does not have to be function, you can follow the examples in `http://www.ee.columbia.edu/~dpwe/resources/matlab/sinemodel/` and copy and paste the commands from the interpreter. Include comments explaining the code and your choice of parameters. Note that none of these effects require more than 3-5 lines of code.

2.1 Effects

1. Roboticize (freddy-robot.wav) and whisperize (freddy-whisper.wav) hw2-vocals.wav using `stft` and `istft` from `http://www.ee.columbia.edu/~dpwe/resources/matlab/pvoc/`

2. Play with LPC formant shifting using the tools from
   `http://www.ee.columbia.edu/~dpwe/resources/matlab/polewarp/`
   (to create freddy-warped.wav).

   (a) Experiment with `warppoles`. See if you change Freddy's gender.
   (b) Vary `alpha`. What effect does this have on the sound? What about on the spectrogram?

3. Play with sinusoidal modeling (`http://www.ee.columbia.edu/~dpwe/resources/matlab/sinemodel/`) and recreate the following effects. Note that you can ignore the residual for most of these effects.

(a) Pitch shift down by 25% (freddy-pitchshift.wav)
(b) Only pitch shift partials above a certain frequency threshold to get a funny effect (freddy-pitchshift2.wav). See if you can make this sound like two people singing in harmony.
(c) Speed up by 25% (freddy-speedup.wav)
(d) Slow down by 50% (freddy-halfspeed.wav)
How does this compare to the OLA and SOLA time stretching from the last assignment?
(e) Quantize the pitch so that the frequencies are always integer multiples of 100Hz in all tracks (freddy-quantized.wav). (This sounds especially cool on this sound.) Describe how you could extend this technique to create an autotune effect.
(f) Tremolo (freddy-vibrato.wav)
(g) Vibrato (freddy-tremolo.wav)
(h) Separate pitched sounds from transients using the analysis based on `ifgram` and synthesis using `synthphtrax`. (freddy-sin.wav and freddy-residual.wav)
(i) Hoarseness (freddy-hoarse.wav)