Homework #4: Laboratory Exercise 4
Due Date: May 17, 2011

# Distortion and Upsampling

**Problem 1. [70 Points]**

In this lab, the plug-in *Distortion* will be explored. The plug-in is posted to the class web site; look for `Lab4_MAC_VST.zip` or similar. Solutions for each exercise should include the source files `Distortion.cpp` and `Distortion.h`, suitably modified. The files must be able to be compiled. Additional write-up is required for some sections of this problem.

The plug-in processing, shown in Figure 1, is a cascade of an input gain and filter, upsampling, distortion, downsampling, and an output gain and filter. The distortion element clips its input to lie within the interval $[-1, 1]$, either with a hard clip,

$$\xi(x) = \begin{cases} 1, & x > 1, \\ x, & |x| \le 1, \\ -1, & x < -1, \end{cases} \tag{1}$$

or with a soft saturation,

$$\xi(x) = \frac{x}{1 + |x|}. \tag{2}$$

By applying parametric sections before and after the distortion element, various guitar amp-like distortion sounds can be made.

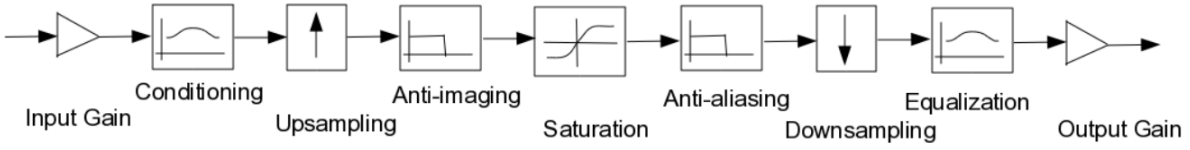


Figure 1: Processing chain

**1(a). [50 Points]** Approximating the distortion element as a Taylor series in $x$, and noting that a signal $x(t)$ raised to the $n$th power has a spectrum $n$ times the bandwidth of the original, we see that the bandwidth of any input to may be greatly multiplied by the distortion element. For this reason, the input signal is upsampled by a factor of eight by inserting seven zeros between each input sample, and then low-pass filtered to 1/8 of its new bandwidth. The 1/8-bandwith signal is distorted and low-pass

filtered again to 1/8 of its bandwidth. One of every eight of these samples is output, the rest discarded.

As presently implemented, the low-pass filter is a fourth-order Butterworth with cutoff frequency of 1/8. It's insufficient to eliminate aliasing. Verify this by setting the input gain to 6.0 dB and the output gain to -6.0 dB, and running the provided linear sine sweep (`linearSweep.wav` inside `audioTracks.zip`) through the plug-in. Plot a spectrogram of the output, and also listen to the output. Turn in your spectrogram and briefly explain what you heard that shouldn't be there if there were no aliasing.

Now design an improved antialiasing filter by using one of the Matlab functions `butter`, `cheby2`, or `ellip`. Adjust the cutoff frequency, filter order (although don't go higher than order 12), and other parameters so that the aliasing is barley noticeable when listening. You will probably need to convert the polynomial coefficients returned by the design function to biquads (sometimes called second-order sections); see functions such as `tf2sos` and `zp2sos`. Turn in your filter coefficients along with a plot of the magnitude of your antialiasing filter.

Do you notice any difference in aliasing in the output when the hard clip is used compared to the soft clip? Why?

Aside from aliasing, do you notice much difference in the sound of the output when the hard clip is used compared to the soft clip?

**1(b). [20 Points]** Implement a parametric section to compute the coefficients of the input and output filters from their controls.

**1(c). [just for fun]** We have posted a couple guitar tracks on the class web site (`Materials` → `Audio Tracks` → `Dry Guitar Tracks.zip`); see if you can't find some filter settings which make a nice distorted guitar sound. Also, come see Dave or Jonathan for information about a tube-like distortion characteristic if you're interested.