ECS730 Digital Audio Effects

Assignment 3: Original Audio Effect

Instructions:

In the language of your choice, implement and evaluate an original audio effect. Your effect should go beyond any of the examples in class, but the content and behaviour of the effect is up to you. You could consider modifying the operation or an effect we've discussed, combining elements from multiple effects, creating an effect in a programming language we haven't covered, or doing something complete original.
Once you've created the effect, create at least two audio examples of its operation on different source material. Also include block diagrams explaining how the effect operates, and figures in the manner of Assignment 1 to demonstrate that your effect works as indicated.
The assignment will be marked in terms of originality, effort and the quality of implementation and operation.

Turn In:

Using the online submission system, submit a ZIP archive containing:

• **Audio files** of your effect in action. Include at least two examples, with both input and output audio files.
• **Commented Code** used to run the effect. Please make your code legible and easy to understand! Please also indicate how your code should be compiled, especially if you use tools other than the provided Juce library.
• **PDF Report**, 3 or 4 sides of A4 total, explaining your effect's motivation and implementation
. Also describe any parameters used to control the effect. Include one or two plots showing the effect's operation and one or two block or signal flow diagrams illustrating how the effect works. You can use Audacity to apply your effect to a recorded file, then analyse the result in MATLAB to generate plots.

Helpful MATLAB Functions: *(use the 'help' function to learn more)*

`[y, fs, nbits] = wavread('filename')` *Read a WAV file from disk*

`wavwrite(y, fs, nbits, 'filename')` *Write a WAV file to disk*

`sound(y, fs)` *Play a vector as sound with sample rate fs*

`plot(), semilogx(), semilogy(), loglog()` *Linear and log-scale plot functions*

`fft(x, n)` *Calculate the FFT of length n. You might want to window the signal first. Also note that the fft is complex. Use* `abs(fft(...))` *to get the magnitude. A discussion of using fft() to generate spectra can be found here:*

http://www.mathworks.co.uk/support/tech-notes/1700/1702.html

`spectrogram(x, window, 'yaxis')` *Plot the spectrogram of x. If window is an integer, x is divided into that many evenly-spaced segments. 'yaxis' places frequency on the y axis and time on the x axis, as it often appears in audio analysis*

`(0:(length(x)-1))/fs` *For a signal x and sample rate fs, go from sample number to time. Useful for the x-axis of time-domain plots.*