POLITECNICO DI MILANO

M. Sc. Music and Acoustic Engineering

Computer Music: Languages and Systems

DISTORTION PLUGIN

Authors' IDs:

10482867, 10521088, 10539533, 10702368, 10751919

Group 5

May 22, 2020

Contents

1 What is the Distortion plugin? 3

2 GUI components 3

3 Symmetrical Distortion 4

3.1 Hard Clipping 4

3.2 Soft Clipping 4

3.3 Exponential 5

4 Rectification 6

4.1 Full-wave Rectifier 6

4.2 Half-wave Rectifier 6

5 Implementation with JUCE 7

5.1 Audio Processor 7

5.2 Audio Processor Editor 8

5.3 Processor and Editor connection 8

1 What is the Distortion plugin?

A distortion-based effect is a non-linear operation used to enrich the typical sounds of an instrument (most used for electric guitars). In this project we implemented some waveshapers, that are distortion effects described by trasformations that relate an output sample with an input sample. This kind of functions are all non linear transformations: they add new harmonic contents in the spectrum (unlike linear functions that modifies just amplitude and phase of the spectrum). In this project we decided to perform 5 different kinds of distortion: Hard Clipping, Soft Clipping, Exponential Clipping, Full-wave Rectifier and Half-wave Rectifier. Each of them has a different characteristic curve, so it gives a different effect on the spectrum of the signal.

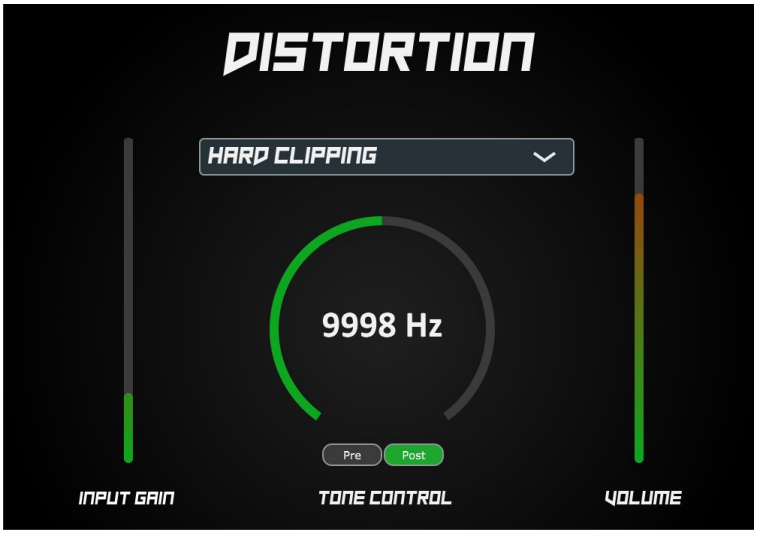

Figure 1: GUI of the plugin

The code used for this plugin is available on GitHub (`https://github.com/Lorenzoncina/JuceDistortionPlugin`).

2 GUI components

The GUI is shown in Fig. 1.The user can choose the type of distortion through the combobox (the drop down menu) and then he can control the amplitude of the input signal with the first slider, that is the input gain. Being the distortion a non-linear effect, the gain of the input signal changes how the effect sounds: the higher the input gain, the higher the distortion. Notice that increasing the input gain does not directly result in a higher volume of the output, since the clipping level remains the same. Instead, to modify the output amplitude the user can control the Volume slider. The Tone Control knob is used to manage the brightness of the sound by changing the cut-off frequency of a IIR low-pass filter.

There are two additional buttons that control the position of the filter in the chain of audio processing: with the 'Pre' option the filter is applied before the distortion effect (this can help reduce the intermodulation distortion by eliminating high-frequency components from the input signal). With the 'Post' option the filter is applied after the non-linear function (this will attenuate the resulting high-frequency distortion products).

3 Symmetrical Distortion

3.1 Hard Clipping

Hard Clipping is a type of distortion effect where the amplitude of a signal is limited to a maximum amplitude. Due to this restriction, the output will never be above a specific level, therefore the input signal is clipped. This is accomplished by detecting when the amplitude of a signal goes above a specified threshold: if this happens, the value of the output signal will be equal to the threshold (resulting in clipping). Assuming the output $f(x)$ is restricted to $[-1, 1]$, it can be represented as:

$$f(x) = \begin{cases} -1 & , Gx \leq -1 \\ Gx & , -1 \leq Gx \leq 1 \\ 1 & , Gx \geq 1 \end{cases}$$

where G is the input gain. As we can see from Fig. 2, this means that the curved top of a sine wave will be flattened down to a straight horizontal line. This will make it look and sound much more like a square wave. The abrupt juncture between the normal waveform and the clipped portion and the high harmonic frequencies can make the audio sound extremely harsh and fuzzy. Hard digital clipping gives the highest apparent loudness, along with the most noticeable distortion and the biggest loss of low bass.

3.2 Soft Clipping

With the technique of Soft Clipping the amplitude of a signal is saturated along a smooth curve, rather than the abrupt shape of hard-clipping. Soft clipping waveshapers produce less distortion and limit the signal in a way that is comparatively pleasant to the ear. The soft clipping we implemented here is based on a quadratic function: this means that in the characteristic curve the linear region and the clipping region are connected by an interval where the output is quadratic with respect to the input.

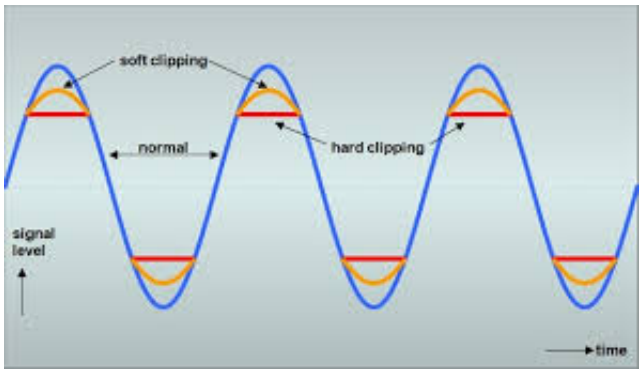

Figure 2: Difference between hard and soft clipping.

3.3 Exponential

Another common way to obtain a soft clipping effect is as follows

$$f(x) = sign(x)(1 - e^{-|Gx|})$$

where, as usual, x is the input, G is the input gain and f is the output. This curve exhibits a quasi-linear behavior for low input amplitudes and transitions very smoothly to an almost flat region for high inputs. A low input gain will give us little clipping and a very slight distortion only for the highest peaks, while increasing it will suppress the exponential term giving us a sharper characteristic curve that has a steeper and narrower linear region and a larger clipping region. The result of this type of soft clipping is a nicely grungy distorted soun where the newly produced harmonics are well balanced and sounding a bit more organic compared to other types.

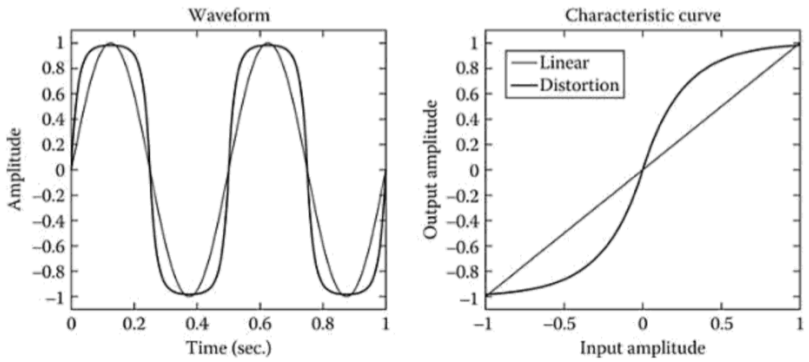

Figure 3: Waveform and characteristic curve for the exponential distortion.

4 Rectification

4.1 Full-wave Rectifier

The full-wave rectified signal is obtained by simply imposing:

$$f_{full}(x) = |x|$$

This, of course, results in an inversion of the negative part of the signal, as seen in Fig. 4. Full-wave rectification introduces a strong octave harmonic: in the case of a periodic input that has no constant offset and is symmetrical with respect to the x axis (such as a sinusoid), the output has a period that is half that of the input, with a complete suppression of the fundamental. This goes to show the strong non-linearity of this effect, that produces a very harsh distorted sound.

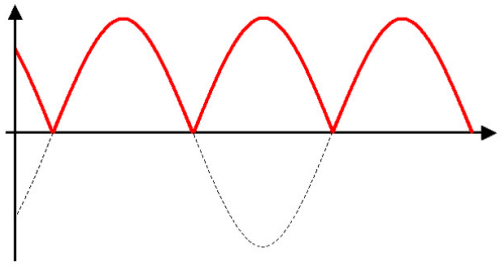

Figure 4: Full-wave rectification.

4.2 Half-wave Rectifier

If the user wants to retain some of the original tone while obtaining a similar effect, they can use half wave or partial wave rectification. With this technique the negative part of the input signal is put to zero, leaving the positive of the original waveform unmodified. The enhanced octave harmonic is still present, but its predominance is obviously less dramatic with respect to the full-wave rectification, and the same holds true for the suppression of the fundamental.

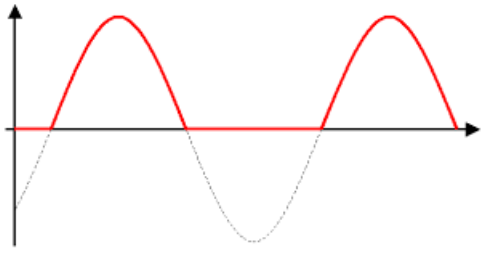

Figure 5: Half-wave rectification.

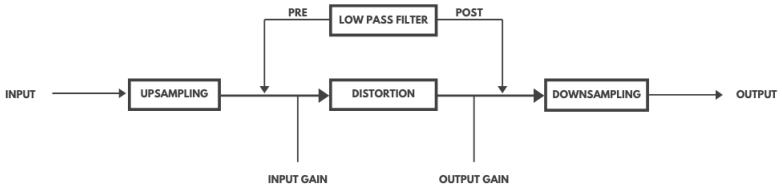

Figure 6: Flowchart of the implementation of the plugin.

5 Implementation with JUCE

This distortion plugin was developed with the JUCE framework in C++. JUCE is a cross-platform framework which allows the export of plugins (or audio stand-alone applications) for different operating systems, relieving the developer from the issue of making their application compatible with the different OSs and allowing them to focus just on the audio algorithm. The code was implemented with the typical workflow for a JUCE audio-plugin, the audio thread is detached from the graphical one by using two different classes: Audio Processor and Audio Processor Editor. Thanks to this operation, priority is always given to the audio thread, which is indispensable for real-time application. The audio algorithm was implemented at first; the chart in Fig. 6 represents the sequence of operations implemented. On top of that, a suitable GUI was designed.

5.1 Audio Processor

The distortion, being a non-linear operation that adds high frequencies, requires the sample rate of the input signal to be augmented (following the Nyquist Theorem), in order to avoid aliasing. After that, the signal is downsampled to the original sample rate. These two operations are performed with the `Oversampling` class of the JUCE DSP module. To perform the Oversampling, an Half Band PolyPhase IIR filter (which minimizes the latency) and a factor of 4 were chosen. Although aliasing was minimized, distortion can still produce a huge number of high frequencies. In some cases these high frequencies can bother and must therefore be mitigated. To do so a Low-Pass Filter was added. The user can control both the cut-off frequency, by changing the values of the central rotary slider, and choose where to place this filter within the audio processing, chain through the two buttons placed below the slider:

- **Pre**: the filter is placed before the distortion. The purpose of this arrangement is to reduce the magnitude of the high-frequency components in the input and, therefore, their con-

tribution to the intermodulation distortion.

- **Post**: the filter is placed after the distortion, decreasing the high-frequency content of the distorted signal.

The Low-Pass Filter was implemented with the `IIR::Filter` class of the DSP module and can be used as a brightness control of the signal, to further customize the final effect. Distortion is the core of the application, obtained with the use of non-linear functions. The choice of the distortion kind to apply, and so the different functions, can be done through the drop-down menu. The amount of distortion to apply is set amplifying the signal by the value controlled with the Input Gain Slider. The audio samples are scaled again by the Output Gain Slider (Volume) to control the final volume.

5.2 Audio Processor Editor

The Graphic Interface was designed using fixed bounds of the elements, because it was decided not to be re-sizable to avoid compatibility issues that sometimes arise between the DAW and re-sizable plugins. Instead of using standard components, the three sliders were completely redrawn from scratch with the JUCE `LookAndFeel` class. For the two vertical sliders, in addition to the shape, a color gradient was added from green to red to advise the user of Gain's increasing value. The 'Pre' and 'Post' buttons were customized too, according to the other components' style. Not showing the dB value of the vertical sliders and place the value in Hertz inside the rotary one was a stylistic choice.

5.3 Processor and Editor connection

The connection between GUI and Processor was implemented exploiting the `AudioProcessorValueTreeState` class that allows to connect with a pointer each graphical component to the corresponding Audio Parameter. Audio Parameters are a recommended choice over simple control variables because they allow to directly draw automation inside the DAW, which would not be possible otherwise. Another improvement that arises from this choice is that it's possible and easy to save the state of the plugin in an XML file.

References

[1] **Will C. Pirkle:** *Designing Audio Effect Plugins in C++*, Routledge 2019, chapter 19 , page 535

[2] **Juce Documentation** *Juce Documentation* (`https://docs.juce.com/master/classes.html`)

[3] **Joshua D. Reiss, Andrew McPherson** *Audio Effects: Theory, Implementation, Application*, Routledge 2014, Chapter 7

[4] **Clipping:** *Major differences between soft and hard clipping* (`https://www.groovesoundesign.com/post/2016-1-20-softclipping-vs-hardclipping`)

[5] **Soft-Clipping vs Hard-Clipping:** *Soft clipping and hard clipping in pedal effects and how they affect your sound.* (`https://humbuckersoup.com/pedals/soft-clipping-vs-hard-clipping-difference/`)