**HW2 - Assignment 4: Distortion effect**

**Github Link -** [**https://github.com/Fra-Boa/Distorsion-Fx.git**](https://github.com/Fra-Boa/Distorsion-Fx.git)

Francesco Boarino - Silvia Messana - Alessandro Molteni

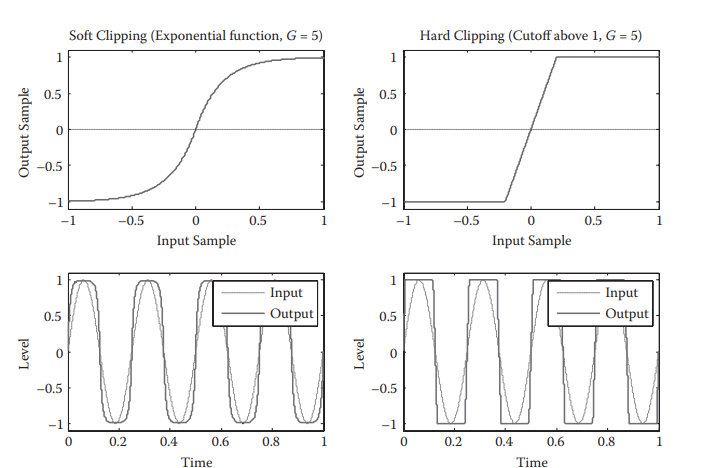
**Introduction**

Distortion is one of the most known effects for electrical instruments. Its first usages date back to the early fifties, when the blues panorama started to feel the need for warmer sounds. Its discovery has been someway “accidental”: when performers raised over the limit the volume of their low-fi amplifiers, they realized that somehow the output sound turned dirtier and “fatter”. Technology behind this phenomenon has been explored and studied, and nowadays we are able to create many possible shades of distortion effects, both in a digital or analog fashion. In this project, we realized a digital distortion effect where the user can control the main characteristics of output sound. The implementation has been done under the Juce framework, a very powerful tool used for the realization of plugins.

**Theory Behind Distortion Effects**

Distortion is realized by applying a nonlinear function to the input signal. As a matter of fact, the chosen nonlinear function influences only *partly* the resulting output. There is another aspect that strongly affects the distorter: the behavior of the output signal when it reaches the maximum of the dynamic range (imposed by the limit of the system it is included in). The condition in which the input signal overcomes the limits of the dynamics is called *clipping*:  when this happens a change in input doesn’t provoke any change in the output.

The way the curve approaches the clipping condition can generate many possible distortion effects, and can be divided in soft and hard clipping: in the former case the curve is smooth and asymptotically tends to the dynamic’s limit, while in the latter one, it sharply reaches the limit and the resulting curve has a edgy aspect.



In most distortion effects, before applying the nonlinear function, an input gain is applied. When it is very high, hard and soft clipping behave similarly: in both cases the output wave approaches a square wave.

The figure on the left shows the difference between hard and soft clipping.

The behavior at clipping level can differ between positive and negative half-waves. Two possible examples, that will be subsequently implemented, are the *half-wave rectifier* and the *full-wave rectifier*.

Half-wave rectifier completely eliminates the negative half-wave, while the full-wave rectifier reverses the negative half-wave generating a distorted wave with a doubled frequency with respect to the original one.

When applying a nonlinear function to the input signal, the spectrum of the output signal can present frequency components that were not present at the input. The two main phenomena that create these artifacts are *harmonic distortion* and *intermodulation distortion*.

The harmonic distortion phenomenon generates frequency components at the output that are multiple of the fundamental frequency, called harmonics. Harmonic distortion has two main consequences:

* If the input signal has more than one input frequency, harmonics of each input will be present.
* The harmonics magnitude decreases as the frequency increases, but it is never zero: this can create aliasing problems.

Generally harmonic distortion is a desirable phenomenon: if the applied function is asymmetrical, the spectrum of the output will present both odd and even harmonics, generating a rich sound.

Differently from harmonic distortion, intermodulation distortion is a phenomenon that musicians tend to avoid. The intermodulation distortion generates frequency components in the spectrum that are linear combinations of the frequencies of the input. The previous statement leads to two general conclusions:

* The complexity of the input is amplified at the output.
* In most cases, between the output frequencies there will not be an harmonic relationship, meaning that the resulting sound will be conflicting and disagreeable.

The negative effects of the intermodulation distortion and harmonic distortion (aliasing) can be fixed thanks to low-pass filters that limit the high frequency components. If the low-pass filter is *before* the application of the nonlinear function, it helps decrease the effects of intermodulation distortion, whereas if it is *after*, it will reduce the high frequency harmonics due to harmonic distortion. The cut-off frequency of the filter is usually user-controlled: this can be seen as a *tone* control, that affects the timbre and the openness of the sound.

There are other important parameters that can be controlled by the user:

* Input gain: as previously stated, it is a simple multiplicative constant applied to the signal before the nonlinear function.
* Volume: gain is normally very high for well-distorted sounds, resulting in an output that is close to the clipping level. In order to rescale the signal, another constant, the volume, ranging from 0 and 1, can be multiplied to the distorted signal.

**Code Structure**

The code structure is divided in three main section:

* Distortion chain
* Filter section
* Additional effects

**DISTORTION CHAIN**

The distortion section is composed by four knobs that drive the amount of distortion applied to the signal (Drive Knob), the range of work of the effect (Range Knob), the percentages of clean and distorted components of the signal that compose the final output (Blend Knob) and the gain (Volume Knob).

The knobs are realized with four rotary vertical sliders, customized through a simple *look&feel* class. Each one of them has a label that specifies the name of the relative parameter.

In addition, we realized a combo box that allows the user to select the proper type of distortion choosing between four different non-linear functions: *Soft Clipping, Hard Clipping, Half Wave Rectifier, Full Wave Rectifier*.

**FILTER SECTION**

The filtering operation is made through a State Variable Filter provided by the DSP module of JUCE.

In the *prepareToPlay()* function we initialize the filter with a *ProcessSpec* object that contains the information about the audio context to use (the sample rate, the number of output channels and the number of samples per audio block).

The *updateFilter()* function is responsible to select the type of filter chosen by the user, and update the value of the cutoff frequency and the resonance.

Graphically, the section is composed by two knobs that drive the cutoff frequency and the resonance of the filter and by a combo box that allows the user to choose between three different types of filter: *Low Pass, Band Pass, High Pass.*

**ADDITIONAL EFFECTS**

The last section is related to two additional effects that can be applied to the final output: a Reverb and a Delay.

The Reverb effect is realized through the *Reverb* object provided by the JUCE framework. Through the *setParameters()* function we set the amount of reverberation applied to the signal (*Dry/Wet* parameter) and the size of the room (*Size* parameter).

The Delay line is realized through a buffer of 10000 samples. The delay operation is made adding at the output stream a copy of the current signal, delayed in time. In this case, the user can choose the amount of effect applied to the signal (*Dry/Wet* parameter) and the time distance between the current signal and its delayed version (*Size* parameter).

Graphically, this section presents two knobs, relative respectively to the *Dry/Wet* and to the *Size* parameters. Furthermore, a combo box allows the user to choose the effect to apply to the signal.

**PARAMETERS RETRIEVING**

The communication between the *Editor* and the *AudioProcessor* components is regulated through a *AudioProcessorValueTreeState* structure, provided by the JUCE framework.

Each of the elements (sliders and combo box) that compose the graphical interface is associated with an attachment, that allows to collect its value in a *ValueTree*. All the parameters are stored in a state of the *ValueTreeState* and are retrieved in the phase of processing of the signal.

**GRAPHICS**

As regards the graphic interface, we uploaded a background and we modified the look of the knobs through a simple *Look&Feel* component, that extracts the snapshots of the custom knobs in the various positions from an image. We also made use of a custom background. All the images are to be included in the JUCE project.

**Choices of Implementation**

In this project we have implemented four different kinds of distortion effects:

* Soft Clipping
* Hard Clipping
* Half-Wave Rectification
* Full-Wave Rectification

Every one of these is easily selectable via the relative combo-box.

In addition to that, we gave the user the ability to use knobs in order to control four main parameters to manipulate the actual distortion signal: Drive, Range, Blend, Volume. (D, R, B, V)

The Drive and Range knobs have the duty of controlling the pre-gain to which the clean signal is subject: both the retrieved values are multiplied together in order to obtain the final Gain value. The first one gives the opportunity to weight the impact of the Range knob (D ranges from 0.001 to 1) which allows it to have a much finer calibration (R ranges from 1 to 3000).

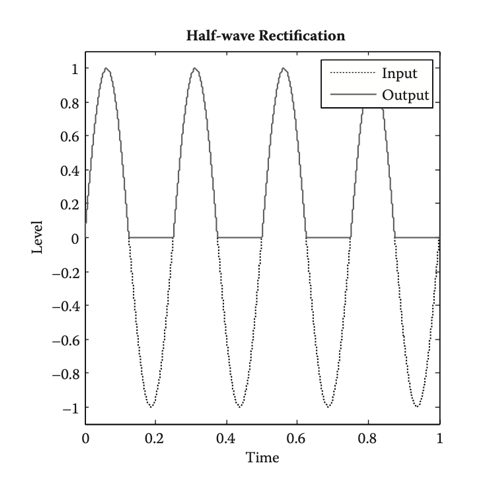
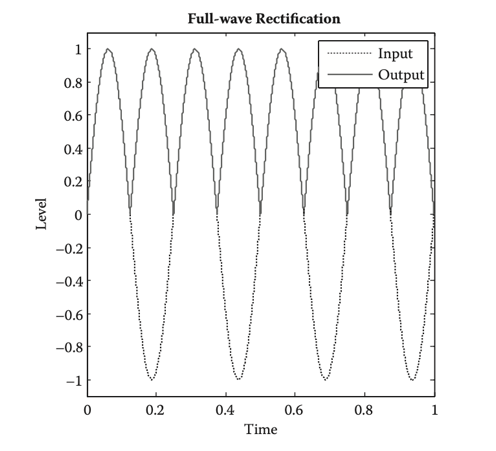
Speaking of the Blend knob, it again gives the user the chance to modify the effect of the distorted signal over the clean one. The higher the Blend value the more impact will have the distortion and vice versa.

At last, the Volume knob is simply a way to control the volume of the blended output signal altogether.

To implement the different kinds of distortion we relied on the mathematical meaning of each one of those.

For the Soft Clipping we used *f* (*x*) = sgn (*x*) ( 1 − *e* −|*G x*| ) where G is the result of the multiplication of D\*R and x is our signal. This allowed us to obtain an output that asymptotically approaches the clipping point as the input gets larger but it never reaches it.

Concerning the Hard Clipping function, we just hardcoded a threshold to 1 and whenever the signal goes beyond that level (or below the opposite in case of the negative half-wave), it simply assumes the threshold level. That allowed to recreate the abrupt transition between clipped and unclipped regions typical of this kind of distortion.

To implement the first Rectification operation, the Full-Wave Rectifier, we just had to take the absolute value of the input signal as: *ffull*(*x*)= | *x* |*.* That allowed to halve the period of the clean signal introducing energy in the octave harmonic which is the main characteristics of the rectification sound.

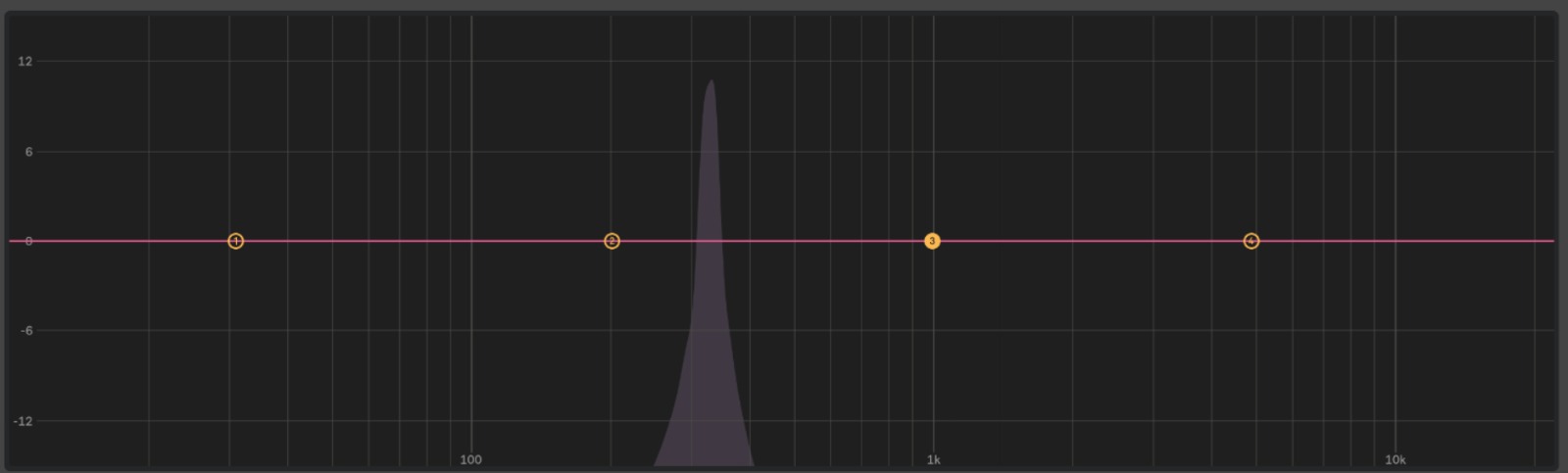
Half-Wave Rectification consists in setting to zero the level of the negative signal while maintaining the positive half-wave. It has been achieved simply by applying: *fhalf* (*x*) = max (*x*, 0).

This operation can also be seen as: *f h a l f* (*x*) = ( *x* + *f f u l l* (*x*) ) / 2 which is the mean between the input signal and its Full-Wave rectified version. For this reason, also this kind of distortion is characterized by having a strong energy component at double frequency with respect to the fundamental.

The last important thing we implemented in order to have a good distortion is the filter. The input clean signal can in fact be filtered before being distorted both to give the user freedom to achieve peculiar sounds and to limit the impact of the high frequency artifacts introduced by the distortion operation in the frequency domain by low-pass filtering.

**Spectral Results**

To have a much deeper understanding of what our plugin was doing to the input audio signal, we have decided to analyze the spectra of the different distorted outputs.



Figure

Figure 1 represents the clean signal with which we have excited our plugin. It is simply a sinusoidal signal slightly over 300Hz.

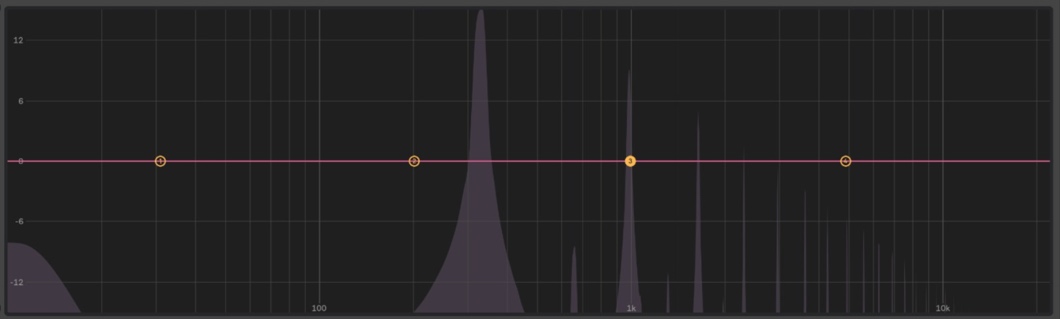
  
Figure 2 is the spectrum after having applied the Soft Clipping function. As we can see, it starts to have energy components in correspondence of the odd harmonics.

Figure 2

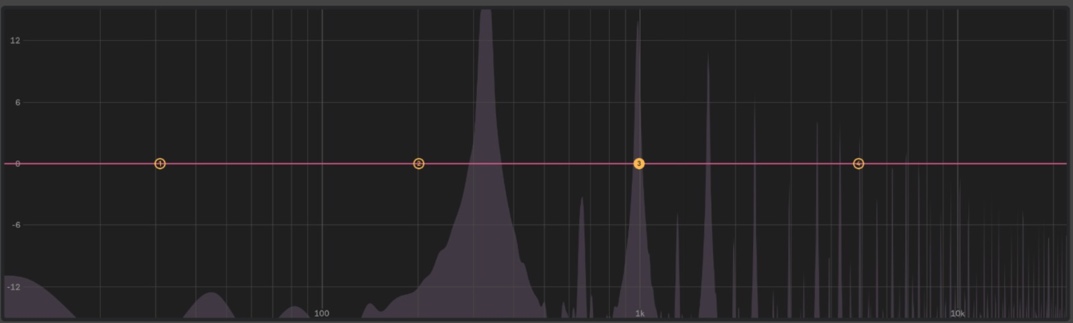
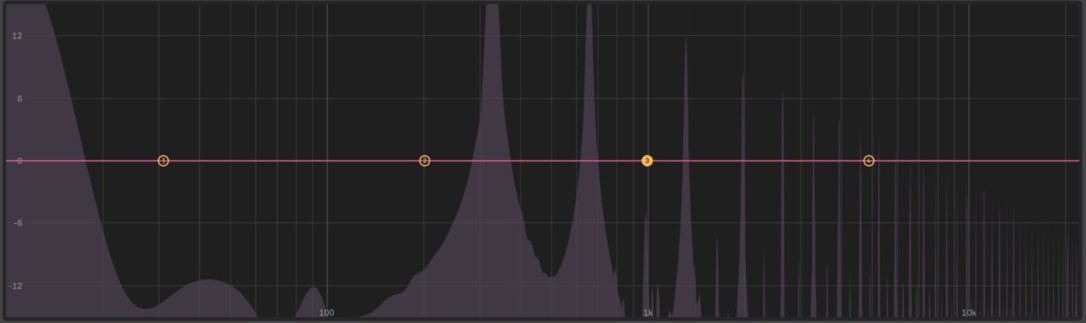
Figure 3 describes the behavior of the spectrum after the Hard Clipping. The result is similar to the one of the Soft clipping but we can see the abrupt “cut” at the fundamental.

Figure 3

Figure 4 is the one related to Half-Wave Rectifier. The main thing we have to notice is the huge amount of energy at the octave due to the halving of the period.

Figure

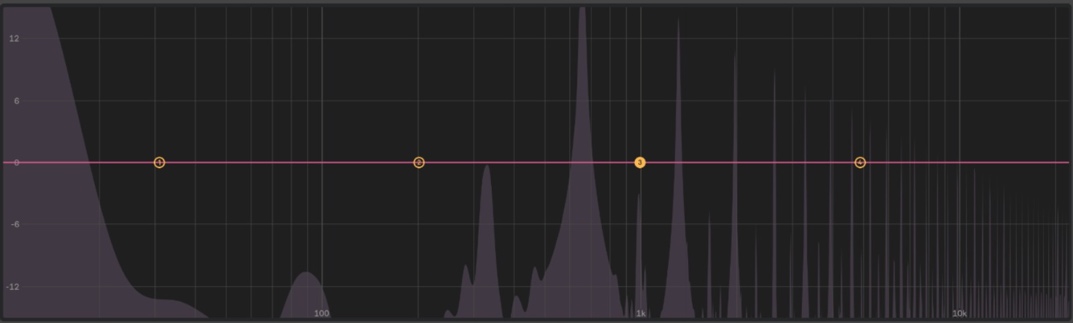
****Figure 5 refers to Full-Wave Rectification. Again we can see how is increased the energy at double the value of the fundamental.

Figure 5

**Conclusions**

What we achieved is a working distortion effect that let the user a good degree of freedom to process his signal. With that been said, as can also be seen from our spectral results, especially in the rectification functions we encountered some issues due to high frequency components generating artifacts. That has compromised the quality of the result of those kinds of distortion. We have tried to solve this problem by Low Pass filtering both the input and the output of the non-linear functions, without obtaining great results.

A future development of the plugin could be trying to resolve these issues by implementing an upsampling/downsampling of the signal, and of course giving even more creative possibility to the user.