Riccardo Alfieri, Riccardo Iaccarino, Manuele Montrasio,
Silvio Sgotto, Francesco Veronesi.

Group Name: *AKGigi* - Group ID: *10*

April/May 2022

1 Introduction

Distortion is a term that is commonly used to define, in general, a nonlinear effect that creates a wide palette of sounds (from smooth and singing tones to harsh and grungy effects) [1].

Any digital distortion effect can be mathematically represented by a function. This function, usually called characteristic function, can be freely designed as long as it keeps the system non-linear, memoryless and time-invariant. This means that the distortion function f must be such that the current output sample $y[n] = f\{x[n]\}$ depends only on the current input sample $x[n]$ and not on any previous inputs or outputs.

In the following document we present how we designed a custom characteristic distortion function in order to create a distortion plug-in with the application framework JUCE.

The report is organised in different sections. The first one covers, in general, how we decided to model the distortion function. In the second section we present how the plug-in has been structured, covering in detail the various components it consists of. In the third section the *Graphical User Interface* (GUI) of the plug-in will be introduced.

2 The distortion function

Distortion effects are often classified by whether they produce *hard clipping* or *soft clipping* [1]. In particular, hard clipping - hard distortion - is obtained when the characteristic equation produces an abrupt transition between the loudest and the quietest regions of the waveform, producing sharp corners in the waveform. In contrast, soft clipping is ensured by a distortion function that creates more rounded corners in correspondence of the waveform peaks. Since, in general, hard clipping produces an unpleasant and overly harsh sound, we preferred to model a function that would generate soft clipping. To do this, we started from the following function - presented in [1] and shown in the first graph of Figure 1 -:

$$y = f(x) = \text{sgn}(x)\big(1 - e^{-g|x|}\big) \tag{1}$$

where g is the *input gain*. As can be seen in Figure 1, the output signal[1] asymptotically approaches the clipping point as the input amplitude gets larger. This particular feature ensures *soft clipping*, since it smoothly reaches the distortion point avoiding abrupt transitions.

[1] As a matter of simplicity, the input signal chosen is a sinusoid, which returns an easily readable spectrum.

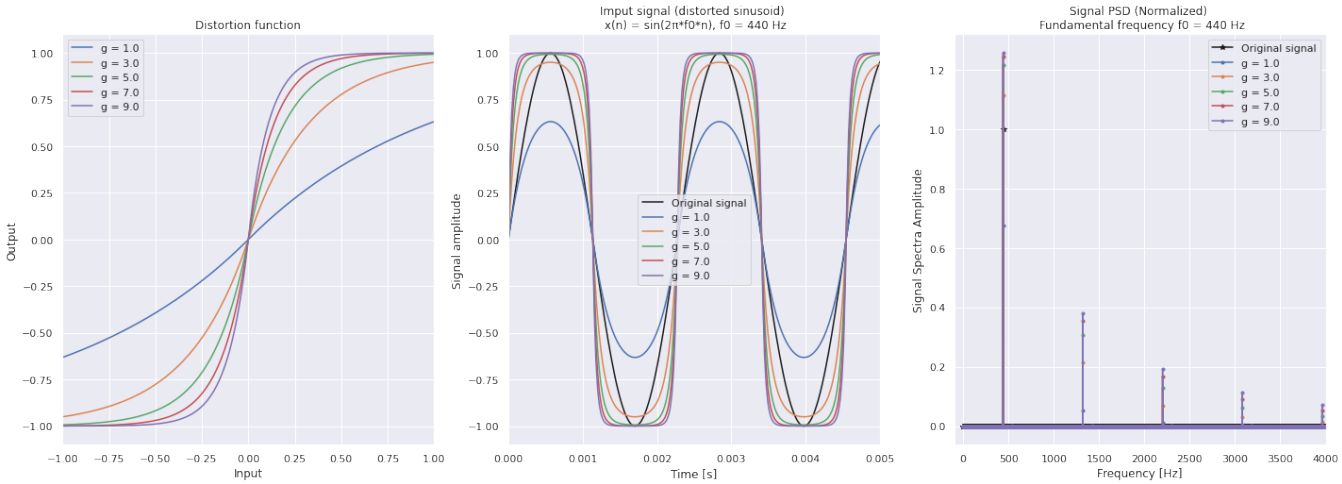

Figure 1: Graphical representation of the main features of the distortion function presented in Equation (1).
First Graph: The characteristic distortion function for different input gains. The function has odd symmetry and ensures *soft clipping*.
Second Graph: Effect of the presented distortion on a sinusoidal signal for different input gains. As the input gain g increases the non-linear effects become more evident.
Third Graph: The normalized estimated power spectral density of the distorted signal. Since the distortion function is odd, only odd harmonics are generated.

Moreover, every nonlinear function will introduce some amount of *harmonic distortion* [1]. The more nonlinear the function, the greater the relative amplitude of the harmonics (see, for reference, Figure 1). In particular, odd symmetrical distortion functions produce only odd harmonics, even symmetrical distortion functions produce only even harmonics, whereas asymmetrical functions can produce both even and odd harmonics.

Musicians often prefer the combination of both even and odd harmonics [1], and consequently asymmetrical functions are often preferred to symmetrical ones. For this reason we decided to modify the distortion function (1), which is odd, so that the symmetry can be varied:

$$y = f(x) = \begin{cases} \operatorname{sgn}(x^h)\left(1 - e^{-g_1|x|}\right), & \text{if } x < 0 \\ 1 - e^{-g_2|x|}, & \text{if } x \geq 0 \end{cases} \tag{2}$$

where g_1, g_2 are the input gains and h is a number that can be 0 or 1. When $h = 0$, the distortion function is always positive, while for $h = 1$ the left part (where $x < 0$) is negative. In general the function defined is asymmetrical, it becomes symmetrical only when $g_1 = g_2$.

In the following graphs it is possible to see how the distortion function affects a sinusoidal signal when the above mentioned parameters vary.

For example, in Figure 2 and in Figure 3 and we can observe the case where distortion introduces respectively a kind of half-wave rectification to the signal and a kind of full-wave rectification. Moreover, in Figure 4 and Figure 5 we can observe the cases where the distortion function is asymmetric and produces both odd and even harmonics.

It should be noted, lastly, that the situation presented in Figure 1 can be recreated by setting $g_1 = g_2$ and $h = 1$.

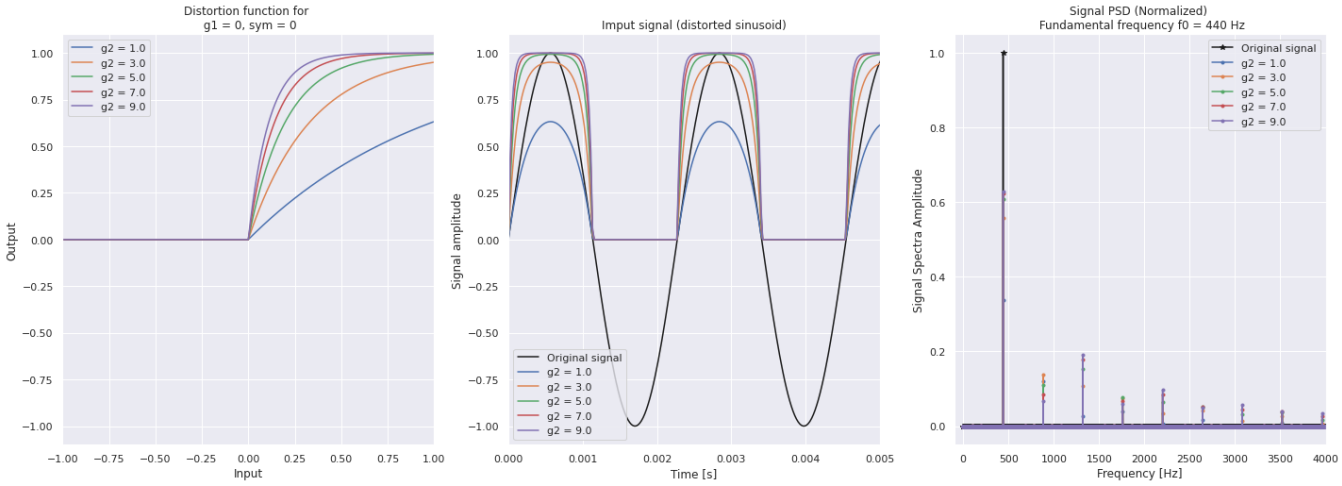

Figure 2: Main features of the distortion function presented in Equation (2) for $g_1 = 0$, $h = 0$ and for different g_2 input gains.
First Graph: The characteristic distortion function is asymmetrical and ensures even and odd harmonics.
Second Graph: The distortion modifies the signal in a similar way that half wave rectification does [1] (sets the negative half-wave to 0).
Third Graph: The normalized estimated power spectral density of the distorted signal. As for half-wave rectified signals, the spectrum contains both the original fundamental frequency and its octave harmonics.

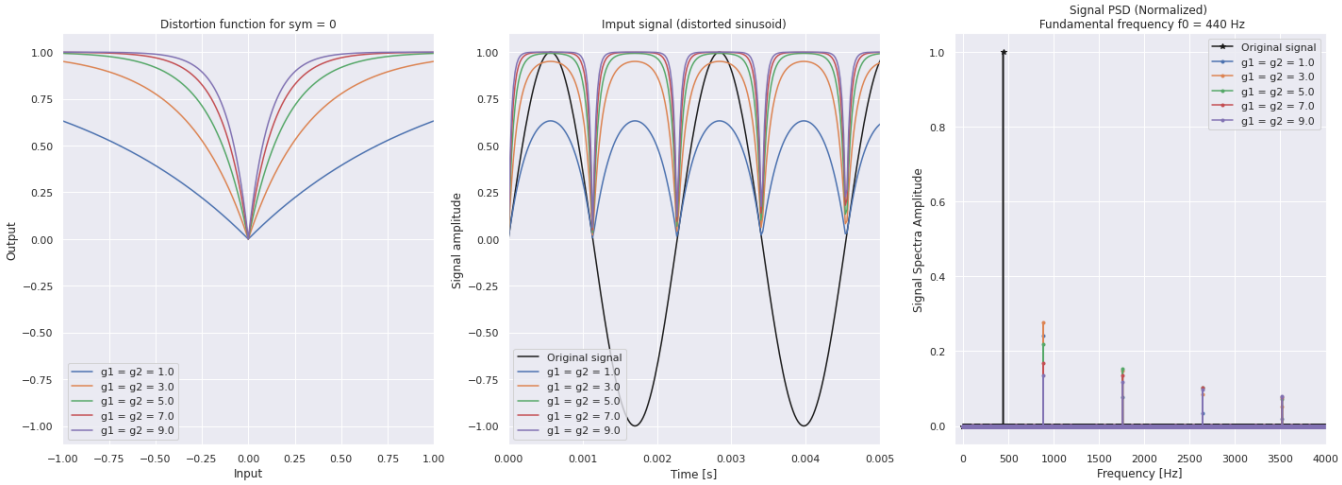

Figure 3: Main features of the distortion function presented in Equation (2) for $h = 0$ and for different $g_1 = g_2$ input gains.
First Graph: The characteristic distortion function has even symmetry and produces only even harmonics.
Second Graph: The distortion modifies the signal in a similar way that full wave rectification does [1] (equivalent to the absolute value function).
Third Graph: The normalized estimated power spectral density of the distorted signal. As for full-wave rectified signals, the spectrum does not contain the fundamental frequency and produces only its even octave harmonics.

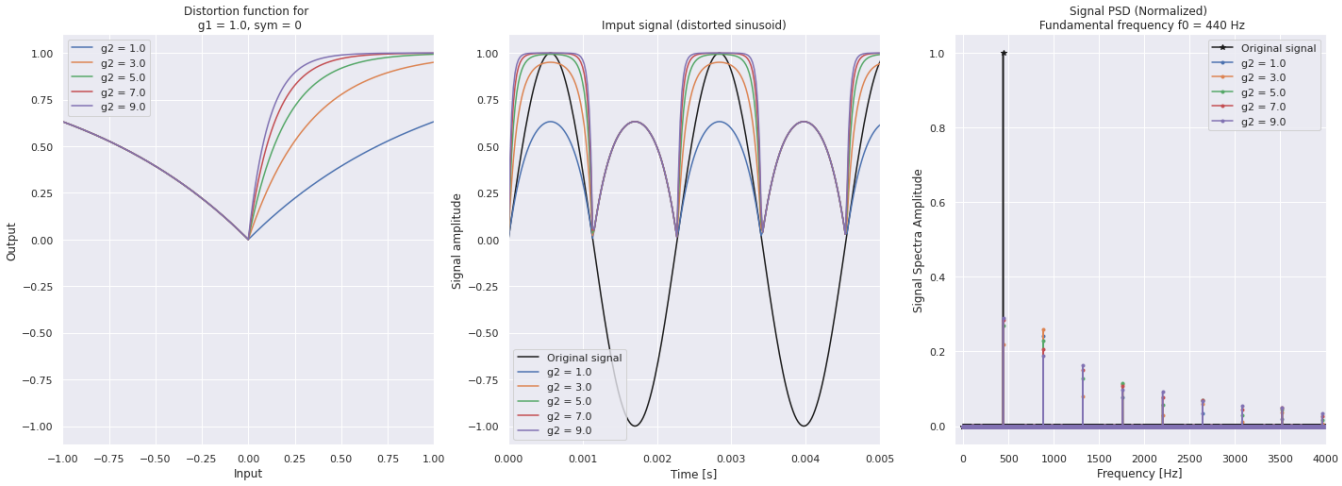

Figure 4: Main features of the distortion function presented in Equation (2) for $g_1 = 1$, $h = 0$ and for different g_2 input gains.
First Graph: The characteristic distortion function has even symmetry only when $g_2 = 1$ and then, in general, produces even and odd harmonics.
Second Graph: As for the case of Figure 3, the signal is always positive but it does not have the typical full-wave rectified shape.
Third Graph: The normalized estimated power spectral density of the distorted signal. The amplitude of the fundamental frequency is significantly lower than in the case of Figure 2. However, it is still present, unlike in the case of Figure 3.

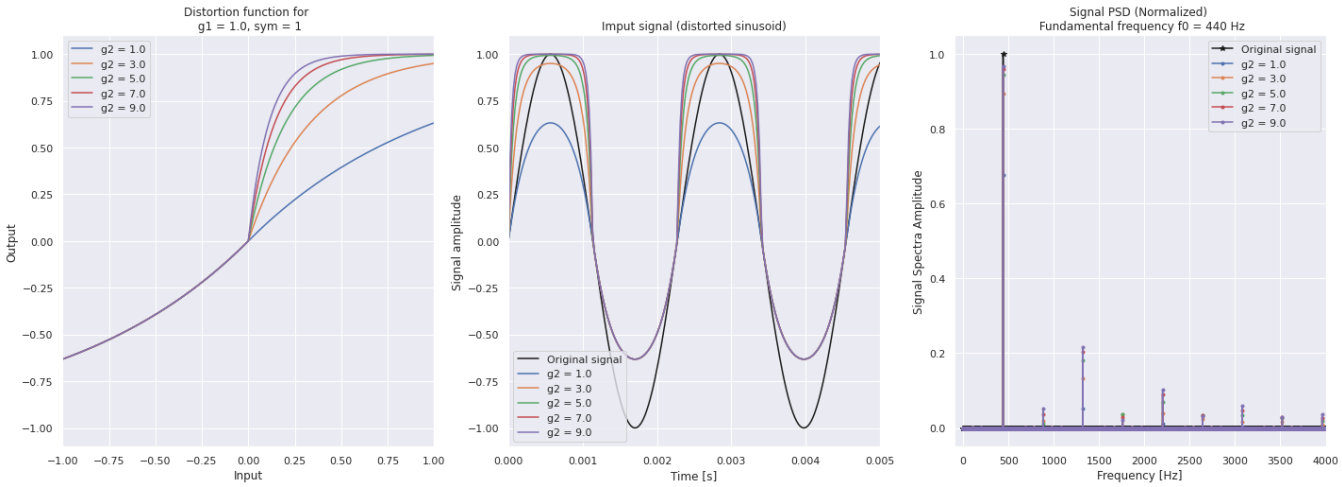

Figure 5: Main features of the distortion function presented in Equation (2) for $g_1 = 1$, $h = 1$ and for different g_2 input gains.
First Graph: The characteristic distortion function has odd symmetry only when $g_2 = 1$ and then, in general, produces even and odd harmonics.
Second Graph: The distorted signal is similar to the one in Figure 1 but only the positive half-wave has a varying shape.
Third Graph: The normalized estimated power spectral density of the distorted signal. The distribution of frequencies in the spectrum is similar to the one in Figure 2, with a wider gap (in amplitude) between odd and even harmonics.

3 The *plug-in* implementation

The *plug-in* that has been implemented, besides adding the previously presented distortion, operates on the signal in different ways. These are schematically presented in Figure 6.

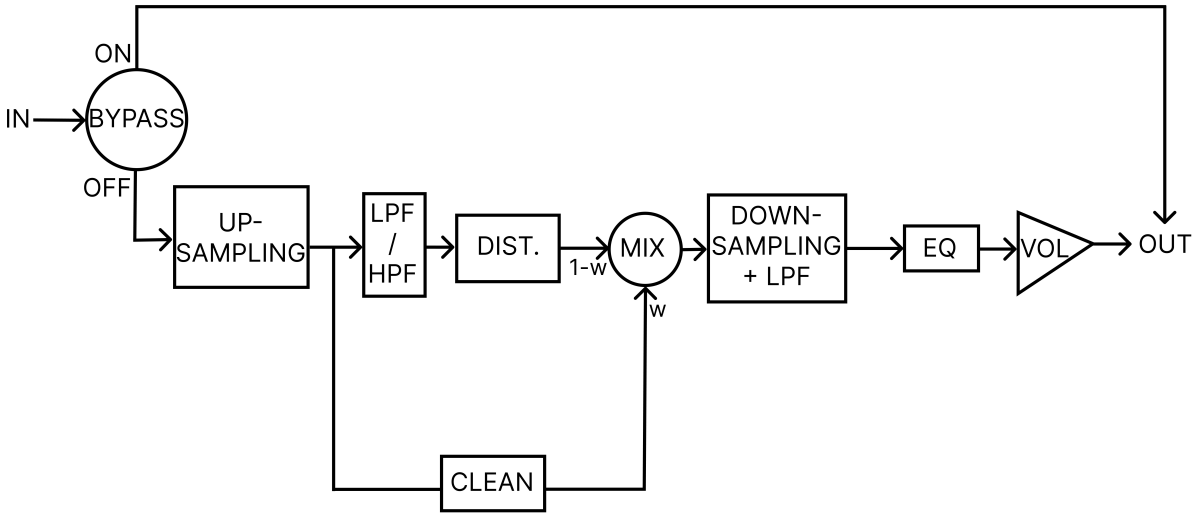

Figure 6: The scheme of the *plug-in*.

- *Bypass*: A must have for every kind of audio *plug-in*. Directly outputs the input signal without applying any effect.
- *Up- and Down-sampling*: Any nonlinear function that is used to create a distortion effect introduces an infinite number of harmonics, frequency components that are integer multiples of an original frequency of the input signal (see, as a matter of fact, Figure 1, 2, 3, 4, 5). In the digital domain, the unbounded series of harmonics creates a problem with aliasing[2]. The best way to reduce aliasing in a distortion effect is to employ oversampling [1]. For this reason, before applying the distortion function, the input signal is upsampled using the *filter Half Band Polyphase IIR* provided by JUCE. Once the distortion is applied, the signal is filtered to remove frequencies above the original *Nyquist frequency* and downsampled back to the original rate.
- *Pre-filtering*: The user can choose to apply a low pass (LPF) or a high pass filter (HPF) to the upsampled signal, also specifying the cut-off frequency (between 128 Hz and 8192 Hz). The former aims to remove the spectral components of the signal which are above the *Nynquist frequency*, limiting aliasing effects and reducing *intermodulation distortion* [1]. The latter has been added to give more expressive freedom to the user.
- *Dry/Wet*: The dry/wet box changes the proportion between the distorted and the non distorted (or *clean*) signal.

[2]Harmonics that are above the *Nyquist frequency* will be aliased, appearing in the output as lower-frequency components.

- *Equalizer*: To further enable the user to manipulate the sound of the distorted signal, we have added a 4-band EQ. Each band covers a different interval in the frequency domain. In particular, we chose to split the hearing range as follows: lows <200 Hz, mid-lows between 200 Hz and 1000 Hz, mid-highs between 1000 Hz and 5000 Hz and high frequencies >5000 Hz. As will be better discussed in the next section, the user can modify the gain parameter of each band.
- *Output gain*: This element allows to scale the level of the final manipulated signal so that it can match the input level.

4 Implementation of the Graphical User Interface

The Graphic User Interface (GUI) that has been implemented for the assignment is shown in Figure 7. The boxes, each one representing one of the previously described elements, are organised in such a way as to show the path of the signal in the *plug-in*, like in a flow-chart.

The starting point is the "input box" (2) from which the signal can go directly to the "output box" (9) through the bypass (1) or continue towards the "sampling box" (3), where the *upsampling*

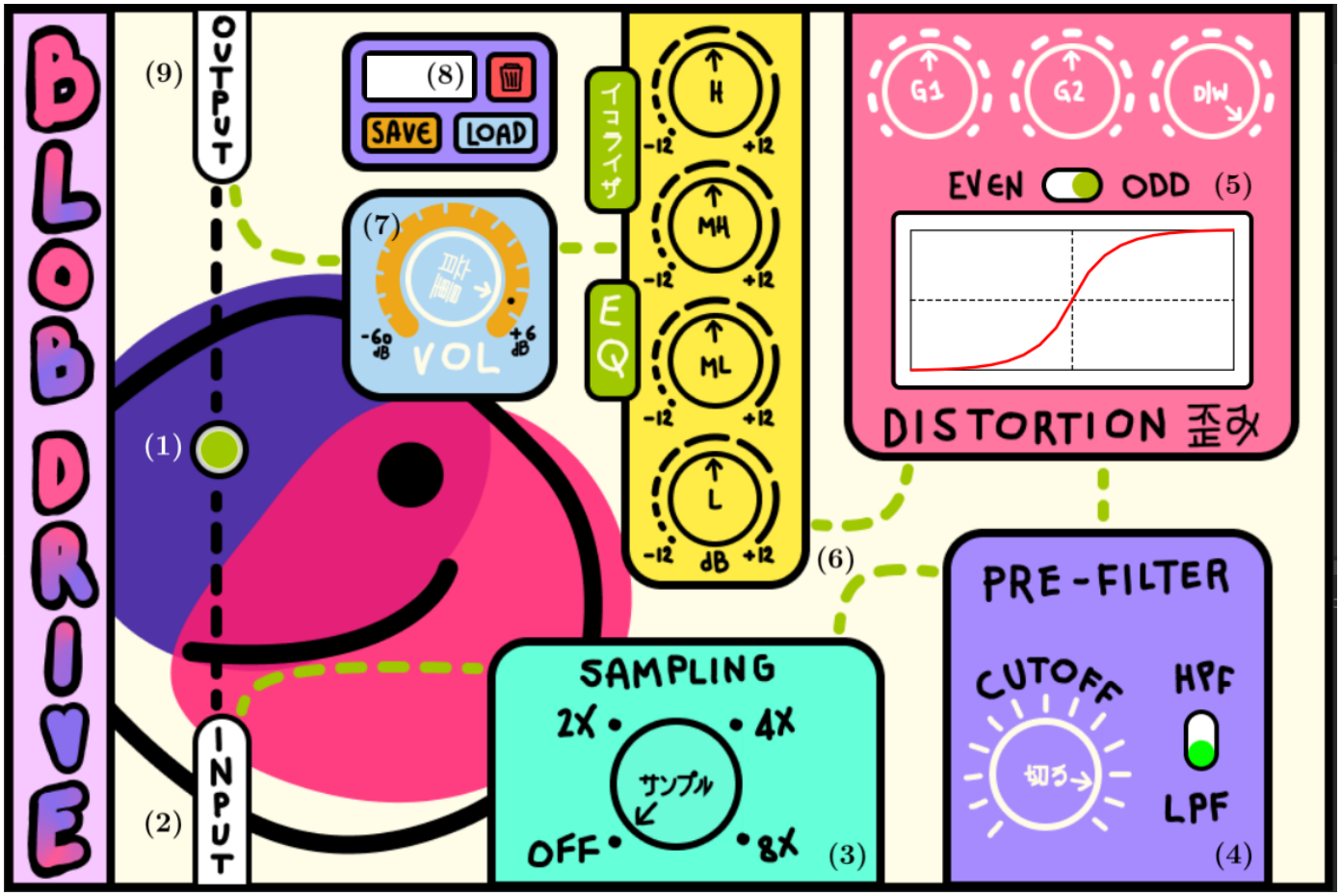

Figure 7: The GUI of the *plug-in*. In order: (1) The bypass button, (2) the input box, (3) the sampling box (where up-sampling happens), (4) the pre-filtering box, (5) the distortion box (with the dry/wet knob), (6) the equalizer box, (7) the output gain box, (8) the save/load box and the output box (9).

factor can be set with a knob. The user can choose between these two paths turning on or off the bypass by clicking on the Blob's eye.

The signal then continues towards the "pre-filter box" (4), in which the user can choose with a switch between an high-pass filter (HPF) or a low-pass filter (LPF) and set the cut-off frequency using a knob.

Right after comes the "distortion box" (5), in which the user can modify the values of $g1$ and $g2$ - as defined in Equation (2) -, choose the dry/wet proportion, decide the symmetry of the distortion function and visualize it in real time in a graph.

In the "EQ box" (6) the user can choose the gain level of low (L), mid-low (ML), mid-high (MH) and high (H) frequencies, using the corresponding knobs.

The last step in the path, before the "output box", is the "volume box" (7), consisting of a single knob that allows to control the output gain of the signal.

Lastly, the "save/load box" (8) gives the user the possibility to upload a default *plug-in* preset, to delete a saved one or to save a new one locally in the `BlobDrivePresets` folder that will be automatically created in the documents folder of the PC.

5 Conclusion

The goal of the assignment was to design a distortion *plug-in* using the JUCE framework. We believe we achieved a reasonably good result, considering the time available and our knowledge. Notably, the distortion effect is not at the same level as the professional ones currently on the market, but we do think we have reached the given "didactic goal".

We are particularly pleased with the distortion function we implemented: it turned out to be very versatile and capable of creating different distortion nuances, ranging from an overdrive distortion (odd symmetry and/or low $g1$ and $g2$) to a fuzz distortion (even symmetry and high $g1$ and $g2$), giving, at the end, great creative freedom to the user. The plug-in's versatility also lies in the fact that it works decently for different instruments, from electric guitar to electric bass and, for the devilishly inclined, also for vocals and drums. For this reason, at least one preset has been created for each of these instruments.

The introduction of other components besides the distortion turned out to be useful to give the user the chance to control the frequency structure of the sound at will, hence affecting its timbre. In particular, pre-filtering and oversampling are fundamental in reducing aliasing, while the dry/wet and the equalizer are designed to personalize the sound. Once again, we are satisfied with the result, although an upgrade of the implementation method used for the various components is always possible.

References

[1] J.D. Reiss and A. McPherson. *Audio Effects: Theory, Implementation and Application*. Taylor & Francis, 2014. ISBN: 9781466560284. URL: https://doi.org/10.1201/b17593.