Audio Programming 2 – Assessment 1

Distortion Plugin Report

Matthew Williams

21479304

University of West London

Lecturer: Peter Dowsett

MSc Digital Audio Engineering – Audio Programming 2

Abstract

With a background in analogue circuitry, distortion is an extremely popular effect to create richer sounds and more interesting timbres in music. To cater for modern digital music production, it is necessary to build on these analogue behaviours in the digital realm.

This report details the background, methodology, and testing of a novel digital distortion effect implemented as a plugin. The effect combines wave folding, bit crushing, and soft clipping to give the user a choice of distortion as well as a combination of all three. The report investigates some background theory behind harmonic distortion and the effects that different functions have on the signal and using this to influence the choices made concerning the parameters and signal flow of the system.

Finally, the report details tests at each stage of the effect and presents conclusions of the plugin and questions what future steps may be worthwhile.

Table of Contents

1) Introduction 1
2) Background Theory 1
2.1) Harmonic Distortion 1
2.2) Wave-Folding Distortion. 1
2.3) Bit Crushing 1
2.4) Soft Clipping 1
2.5) Plugin Process 2
3) Methodology 2
3.1) Distortion Class 2
3.2) Parameter Mapping 2
4) Testing and Results 3
4.1) Wave Folder Test 3
4.2) Bit Crusher 5
4.3) Soft Clipper 6
4.4) Combination 8
5) Conclusions 8
6) References 9
7) Appendices 9

List of Figures

Figure 1 - Signal flow of the System
Figure 2 - Output of Wave Folder to the Same Input with Different Thresholds
Figure 3 - Normalised Frequency Response of Wave Folder to Different Thresholds
Figure 4 - Offset Adjusting to Zero Value Input
Figure 5 – Frequency Response to Wave Folder with +0.5 Bias
Figure 6 - Bit Crusher Input/Output at Different Step Sizes
Figure 7 - Frequency Response of Bit Crusher at 2 Steps and 8 Steps
Figure 8 - Output of Soft Clipper at Different Alpha Values
Figure 9 - Frequency Response of Soft Clipper a Different Alpha Values
Figure 10 - Soft Clipper Output and Frequency Response to Different Negative Thresholds
Figure 11 - Output at Each Distortion Stage and Summed Output

1) Introduction

Distortion is one of the most widely used effects in music production. With its roots in the non-linear behaviour of analogue circuitry, distortion has become an incredibly popular effect in modern music. With the vast majority of modern music being produced using digital equipment, it is necessary to translate the behaviour of analogue circuits to digital systems. This report details the process of creating a novel digital distortion plugin.

2) Background Theory

2.1) Harmonic Distortion

Distortion works by applying a non-linear curve to the amplitude of a signal, enabling the wave to be shaped by the function being applied to it.

Distortion effects with odd functions applied to them will result in only odd harmonics being generated. For both odd and even harmonics to be generated, the function must be neither odd nor even. For only even harmonics to be generated, the function must be even. Therefore, to add a mix of harmonics, the signal must be shaped differently for positive and negative input values.

2.2) Wave-Folding Distortion.

The core theory behind a wave folder is that the wave is reflected over a threshold value, causing it to be inverted about the threshold. Depending on the design choice, the wave may be folded about the threshold many times resulting in richer timbres.

A parameter that is often implemented to a wave-folding system is an offset which shifts the input up or down. This adds even harmonics to the output signal by shifting the distortion function and removing its odd symmetry.

The method used makes use of Simon Hutchinson's (2021) parameters and wave folder layout. The find the output each folding stage uses the equation:

$$y(n) = \begin{cases} -x(n) + 2\sigma\,, & x(n) > 0 \\ -x(n) - 2\sigma, & x(n) < 0 \end{cases} \qquad [1]$$

Where σ is the threshold.

2.3) Bit Crushing

A bit crusher distortion takes a signal and quantises the amplitude to a user determined number of steps. This removes a large amount of the fidelity but adds an aggressive sound as a result of quantisation error. As the number of steps is increased, the amplitude of the higher frequency harmonics decreases until eventually the steps reproduce same output as the input.

The method used makes use of the round function as provided by Eric Tarr (2020) whereby the signal has a gain coefficient equal to the amount of steps required, then is rounded to the nearest integer, and then attenuated to normalise the signal to its original amplitude.

2.4) Soft Clipping

Soft Clipping works by applying a smooth gain function to the signal. To achieve this smoothened gain curve, equation 1, provided by Will Pirkle (2013), is used:

$$y[n] = \frac{a}{tan^{-1}(k)} \cdot \tan^{-1}(kx[n]) \qquad [2]$$

Where a is the threshold and k is the alpha value used to change the steepness of the curve. This creates an effect similar to analogue tube amp overdrive.

2.5) Plugin Process

The distortion effects are applied in series and in parallel with different gain coefficients. The signal flow of the system can be seen in figure 1.

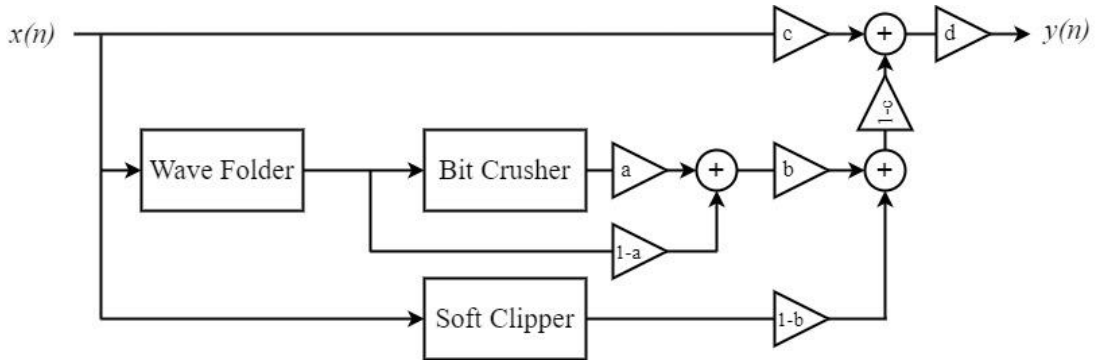

***Figure 1** - Signal flow of the System*

The input signal $x[n]$ is split into three paths, one going to the wave folder, one going to the soft clipper, and the other remaining dry. The signal out of the wave folder is then split into two paths. One is sent to the bit crusher and the other is then summed with the bit crusher output. This creates a mix between the folded signal and a quantised version of the folded signal. This is then summed with the output of the soft clipper to create a mix of the three effects.

3) Methodology

3.1) Distortion Class

A class is used to store the distortion algorithms and variables. To change and access the variable values, public *getter* and *setter* functions are used. The *getter* returns the value of the variable, and the *setter* changes the value of the variable. This ensures that the main program is able to access the variable values but not access the variable directly.

Alongside the variables in the private members are the functions to carry out the distortion effects. These functions are then combined in a public distortion process function. By keeping the distortion processes private it not only secures the programming of the distortion, but it also makes the code in the processor much cleaner and easier to debug.

3.2) Parameter Mapping

The parameters are controlled by sliders on the screen. The *Slider* class is used to create sliders with a range of optional parameters to adjust the operation. The sliders are mapped to parameters using a *ValueTreeState* in the plugin processor. This allows you to store multiple *AudioParameter* classes of differing data types and reference them through the value tree. There is *AudioParameter* for each slider which corresponds to a variable in the distortion class. In the plugin editor, a pointer of type *SliderAttachment* connects the value from slider to a corresponding *AudioParameter*.

To send the *AudioParameter* values to the Distortion Class, a *SliderListener* is used which detects when a slider's value has changed. Using a series of if statements, the specific slider is identified and the *AudioParameter* value is sent to the processor to be updated.

In the processor, each parameter has a *current* and a *previous* variable to allow for parameter smoothing. Parameter smoothing creates a ramp so that the value of a parameter gradually changes to the slider's value, removing audible artifacts such as clicks and pops that may occur when a parameter is changed rapidly. To implement this, a simple first-order low-pass filter is used which requires the current value, which the parameter ramp is changing to, and the previous value, which will be added to the current value at a high previous value to current value ratio. The output from the parameter smoothing is sent to corresponding setter in the Distortion Class.

To set the bias for the wave folder, the offset needs to be set to zero when there is a zero value at the input to ensure that the output is not a constant DC value. To do this, the value of each sample is saved in a float variable in the process block. This will then store the final value of the buffer once the processing has completed. An if statement is then used before the buffer to determine if the input is zero and if it is the *current* offset is set to zero, otherwise it is set as the value of the slider. This means that the offset can be set to zero for a zero input, however, the slider stays at what the user set it at.

4) Testing and Results

Each distortion effect will be tested independently followed by a combination test. The tests involve using a sine wave and analysing the output in the time domain and frequency domain.

4.1) Wave Folder Test

To be judged successful, the wave folder needs to apply the inverse of the wave about the threshold. The wave folder has 4 folding stages so needs to 'fold' 4 times about the threshold if the input has a high enough amplitude to do so.

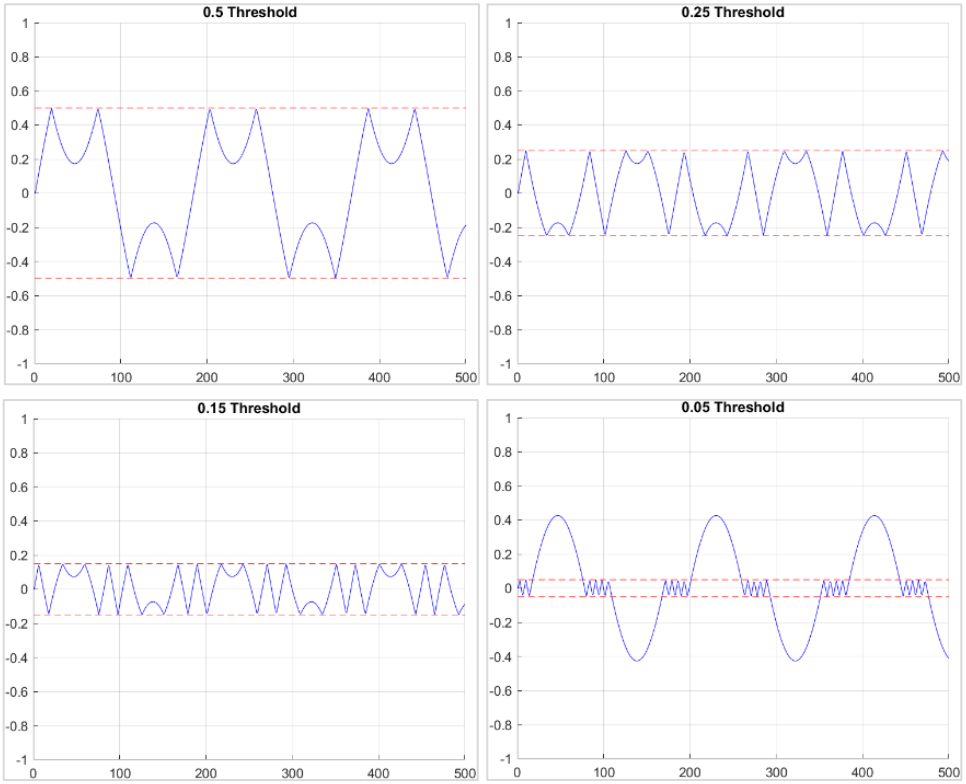

***Figure 2** - Output of Wave Folder to the Same Input with Different Thresholds*

Figure 2 shows the output of the wave folder at various thresholds. The threshold value is superimposed for reference. As can be the seen, the wave folder successfully applies the inverted wave about the threshold with each figure displaying an additional fold from the previous. Figure 2d displays the maximum number of folds being 4 in the positive and negative with the resulting excess wave having an amplitude above the threshold.

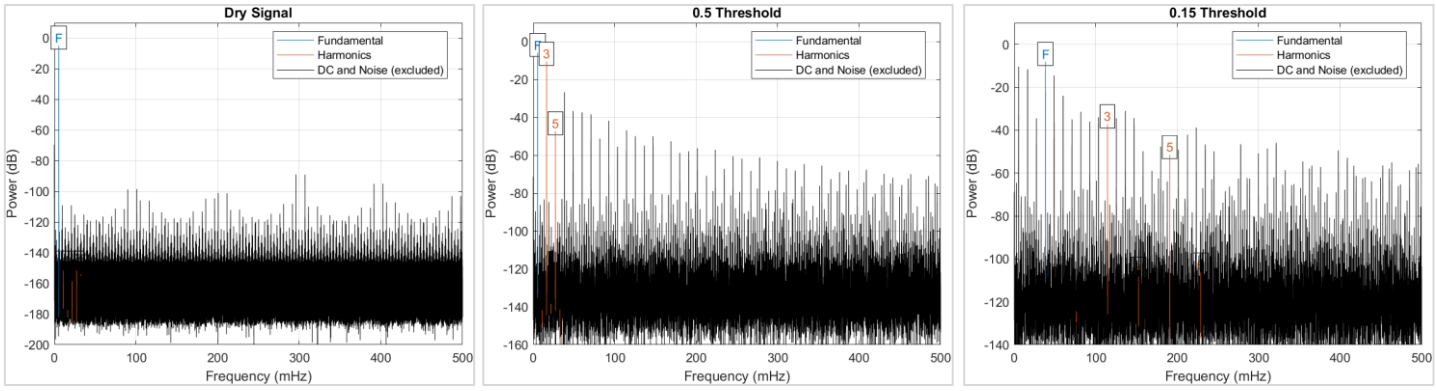

***Figure 3** - Normalised Frequency Response of Wave Folder to Different Thresholds*

Figure 3 shows the frequency response of each wave fold threshold. It can be seen that the wave folder creates odd harmonics in the spectrum with a fairly steady decrease in amplitude for a threshold of 0.5. For a threshold of 0.15, the fundamental frequency no longer has the highest amplitude in the spectrum. Additionally, the higher frequency harmonics have a higher amplitude across the spectrum, trailing off in a less stable fashion.

To test the bias parameter, a sine wave with silence at the beginning and end of the sample was inputted to test that the output is zero for a zero input. Figure 4 show the output of the plugin with a DC bias of 0.5 applied. Successfully, the bias is only present for the values that

are not zero with the parameter smoothing curve being displayed at the end of the signal as the output is set to zero. The signal also shows that the parameter smoothing curve is non-linear and takes approximately 2 seconds from being given a new *current* value and reaching the value.

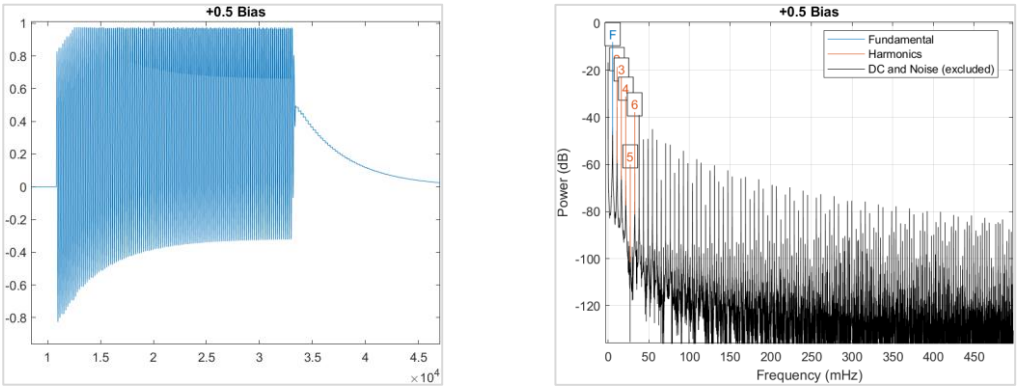

***Figure 4** - Offset Adjusting to Zero Value Input*

***Figure 5** – Frequency Response to Wave Folder with +0.5 Bias*

Figure 5 Shows the frequency response of a signal with a bias of +0.5. It can be seen that unlike the signals with no bias applied, there are even harmonics present in the output.

4.2) Bit Crusher

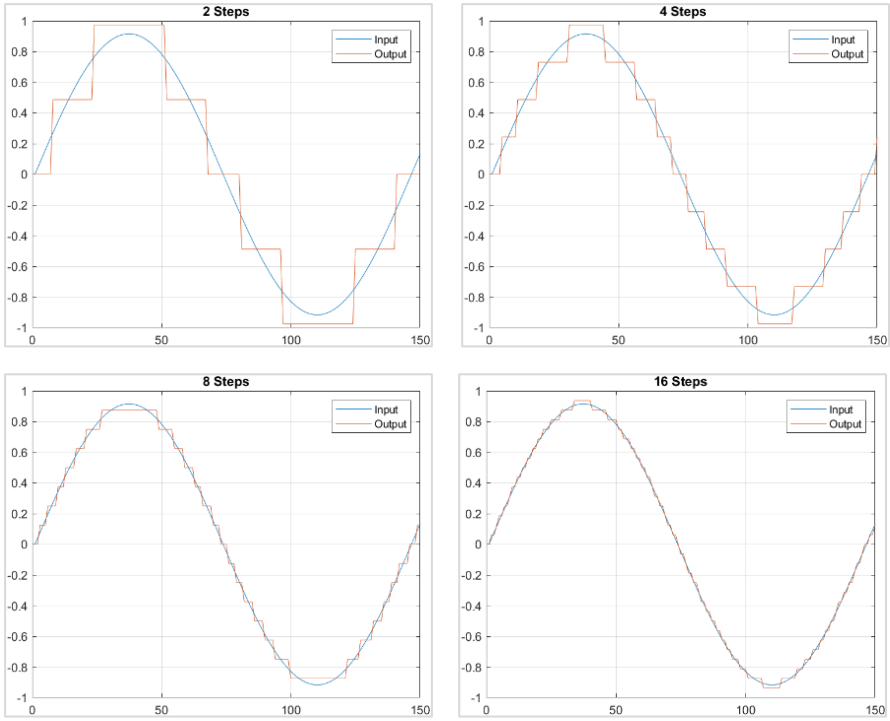

***Figure 6** - Bit Crusher Input/Output at Different Step Sizes*

The bit crusher only has one parameter to control which is the number of steps that the signal is being 'crushed' down to. The number controls the number of steps either side of zero. This means that the actual number of values the signal can take is $2\alpha + 1$ where α is the number of steps chosen. These different step values can be seen in figure 6. It can be confirmed that the distortion effect does as intended as each value is quantised to the number of steps

required. Also, by superimposing the input signal it indicates that the values have been rounded to the correct steps.

Figure 7 shows the frequency representation of 2 steps and 8 steps. Odd harmonics are present in the signal and have a nearly flat roll-off. It can also be seen that the bit crusher introduces a large amount of inharmonic content which is perceived as noise to the listener. This noise floor reduces as the amount of steps is increased as can be seen in figure 7b.

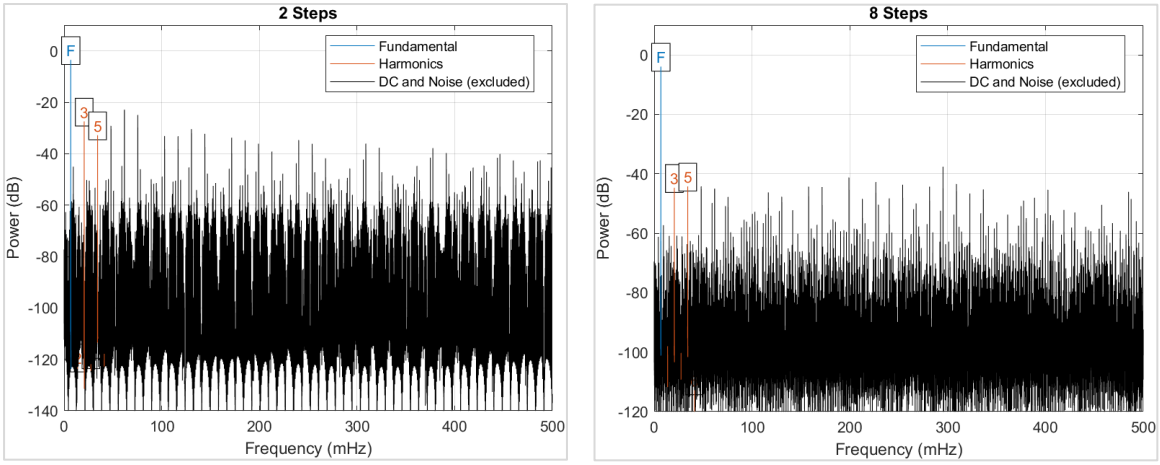

Figure 7 *- Frequency Response of Bit Crusher at 2 Steps and 8 Steps*

4.3) Soft Clipper

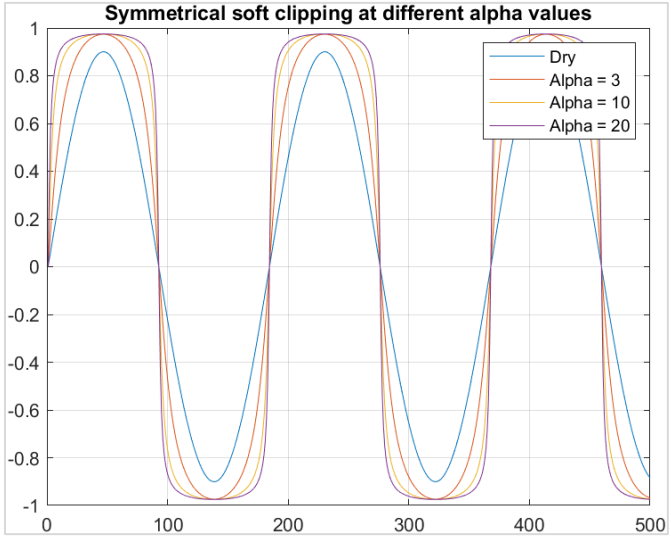

Figure 8 *- Output of Soft Clipper at Different Alpha Values*

Figure 8 shows the output of the soft clipper from an input sine wave at different alpha values. The increase in the alpha value increases the amplitude of the signal and changes the shape to be more square-like. The effect of this can be seen in figure 9 which displays the frequency representation at an alpha of 3 and 20. They both contain only odd harmonics and the high alpha flattens the curve at which they decrease in amplitude, introducing more high frequency harmonics to the signal.

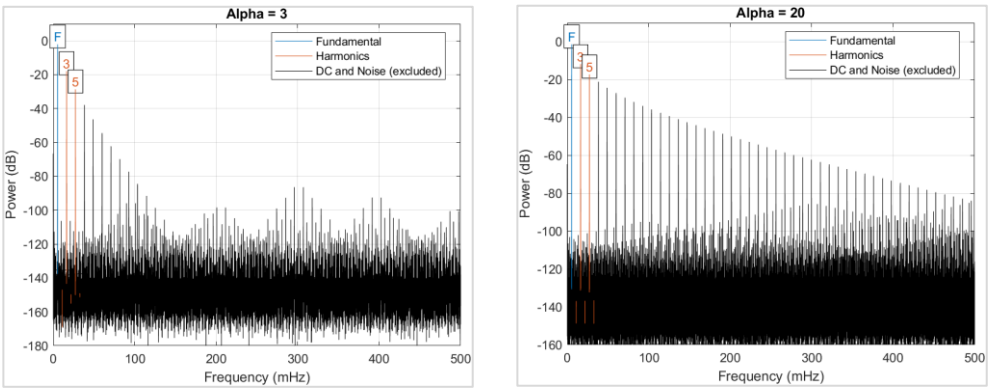

Figure 9 *- Frequency Response of Soft Clipper a Different Alpha Values*

The plugin's asymmetric parameters have been tested which can be seen in figure 10 where the alpha value and thresholds are different for positive and negative values. Although not specifically stated, this gives the plugin a rectifier distortion effect too. The effect of this rectifier distortion can be seen in the frequency responses which introduce even harmonics to the signal. A lower negative threshold causes the odd harmonics to trail off quite steeply. With a fully rectified signal the frequency response shows that only even harmonics are added to the signal.

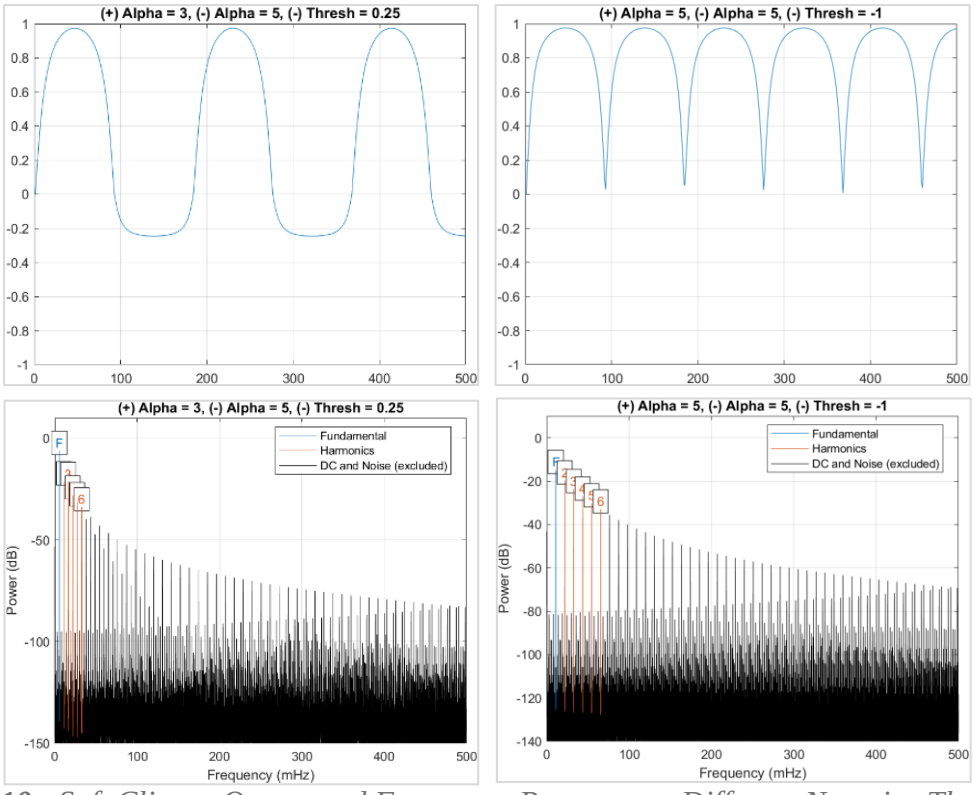

Figure 10 *- Soft Clipper Output and Frequency Response to Different Negative Thresholds*

4.4) Combination

Beyond reasonable doubt, it can be confirmed that the individual distortion effects work in their intended way. The effects are combined through the summing of each signal path by coefficients that add to equal 1 in a wet-dry mix. Figure 11 Shows each stage's output with arbitrary parameter setting followed by the combined output of the plugin. It can be seen from the figure that distinct features of each stage are present within the final output of the plugin confirming that output is a result of summing the previous stages.

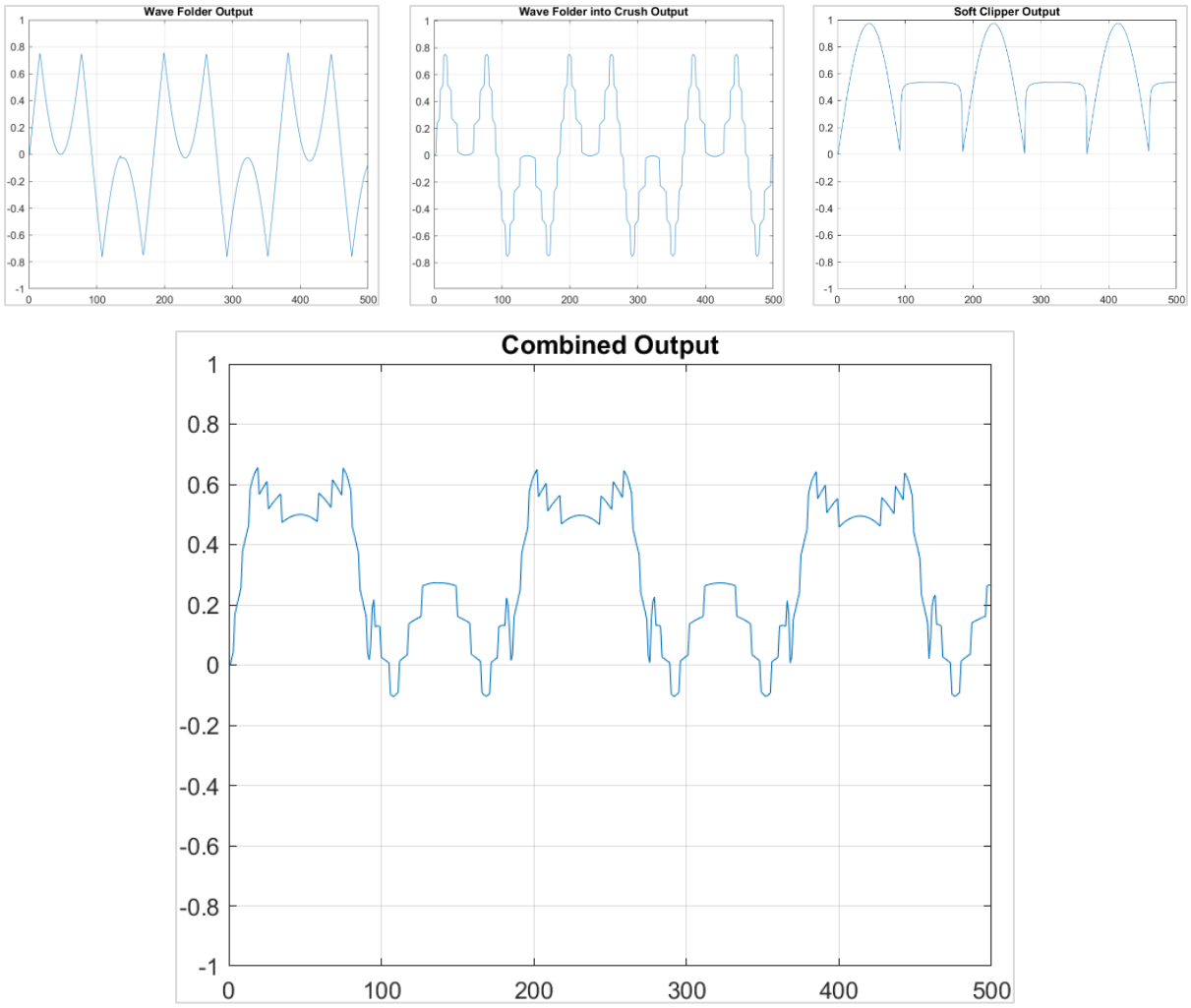

Figure 11 *- Output at Each Distortion Stage and Summed Output*

5) Conclusions

In conclusion, the research conducted has resulted in the creation of a novel distortion effect using different parallel and series stages of previously existing distortions. The plugin successfully shapes the wave in a non-linear fashion to create new harmonics in the sound. The distortion effects all work as they are intended, as proven through analysis on MATLAB. As a result, the previously explored DSP theory is apparent in the testing and proves that the distortion effects work through the theory behind harmonic distortion.

The plugin contains two main signal paths which serve different sonic purposes. The top signal path creates a ‘fuzz’ sound with many high frequency harmonics dominating the signal. The bottom path generates a more typical overdrive effect with a rounded-square type wave. The effects, in combination, allow the user to utilize both types of distortion allowing them to create a ‘full’ sound from the overdrive with the added fuzz of the bit crusher and wave folder.

The background research has helped to inform decisions made concerning parameter choices to allow the user to control the shaping of the signal. In the future perhaps the use of a ‘black box’ style parameter system could be explored whereby the user is able to control a number of parameters through one control. The implementation of this type of control is one of the benefits of using digital systems and it would reduce the number of sliders on the screen, allowing the user to be able to change the timbre with ease.

6) References

Hutchinson, S., 2021. *Making a Wavefolder in Reaktor 6 Primary | Simon Hutchinson*. [video] Available at: <https://www.youtube.com/watch?v=3zPa7Ri276U&t=663s> [Accessed 4 December 2021].

Pirkle, W., 2013. *Digital audio effect plug-ins in C ++*. New York: Focal Press, p.497.

Tarr, E., 2018. *Hack audio*. London: Routledge, Taylor & Francis Group, pp.147-180.

Tarr, E., 2020. *Bit Crushing - Hack Audio*. [online] Hack Audio. Available at: <https://www.hackaudio.com/digital-signal-processing/distortion-effects/bit-crushing/> [Accessed 4 December 2021].

Zölzer, U., 2011. *DAFX*. Chichester: Wiley, pp.115-132.

7) Appendices

Appendix 1:
Link to Github repository:

https://github.com/mattwilliams2099/DistortionPlugin.git