Nonlinear Signal Processing

Theory and Models in view of Audio Systems Emulation

Guglielmo Fratticioli

A thesis presented for the degree of
Doctor of Informatics Engineneer

Informatics Engineering
Univerità degli Studi di Firenze
Italy

Content

Abstract

In this thesis the topic of Nonlinear System Modeling is addressed, it will be followed an approach which works well when emulating nonlinear audio distortion devices. In first the necessary theoretical knowledge is presented, then a specific case will be examined in order to show an operative method to actually model a system.

This methods follows what has been written in [1][2][3]; my main goal is to give an exhaustive explanation of those article to make them easier to digest to novices.

1 Background

In order to explain the reason that brought me to research on this subject matter not only I have to refer to my learning path of the past 3 years but it's worth mentioning the interest that has grown in me regards computer music.

Well if I have to start from that we have to go back to my high school's last year. Actually at that time I was pretty much into starting a carreer at Physic's University (we can say I was fascined by the study of natural systems throught formulas' and mathematics). However a breakthrough happend, I started looking into Electronic music (which around 2015-2019 had been growing in popularity) but that's not all, it could have been for the fact that I have been playing the piano since childhood or whatever, eventually I started learning how to actually produce e-music on my own laptop. That made me realize on thing: I was fascined by the software tools that could generate and process sounds.

Inevitably that was reflected on my university career as I enrolled the course of Informatics Engineering in my hometown, Florence. I had the chance to learn the basics of coding, software engeneering and the essential mathematical tools to understand how to approach this kind of subjects. It's worth noting the importance of some courses as ' Foundaments of Signal Processing ' , 'Numerical Signal Processing ' which has well explained to me important concepts as Signal Analysis, Processing , filtering and how to formally express the related theory.

Then it came the time to choose a topic for the final graduating exam, of course I was oriented on working on something related to Sound Processing and Analysis. The major interest in this area are : Analog Circuits Modeling, filter design (EQ, denoising), dynamic processors (amplitude compressors), spatial effects (reverberation, stereo imaging), the last but not the least, distortion devices emulation (waveshapers,saturators, overdrives). As the electronic music is characterized by an heavy use of distortion processing of the sounds, and given the fact that emulating nonlinear high quality distortion devices in the digital domain is far from being an easy problem, I have choosen to challenge myself in such field.

1.1 On Audio Distortion

As I have mentioned the problem of Audio Distortion is not a straightfoward one, as several approach can be taken to get to similar results. The main goal is to develop a processor that changes a sound producing a distorted version of that. Speaking about Audio, with 'distortion' we usually refear to a waveshaping effect in time domain of the input that generates an output richer in harmonics (higher frequencies). in one hand if it's applied subtly it makes the sound sonically more present and pleasing to the ear. on the other hand a strong distorting system find its use as it's a way to heavly change the timbre as an artistic choice (you can think of the distorted guitar sound). Moreover certain nonlinear behaviours of analog circuits has been found to distort the sound in a more 'musical' and 'pleasing' way than others, that's why a lot of efforts has been made to find a way to emulate them with computers; that's also the main purpose of this essay, to give an operative method to digitally model a nonlinear system in a black box approach.

2 Theory

In the following passage it will be presented an overview on the essential thoretical concept to be known in order to understand this thesis expressed in mathematical formalism.

2.1 Signal Theory

The meaning of 'signal' and 'processing' is what I'm beginning with, we can say a **signal** is the trace of a certain property which is varing in time or space, so in the specific case of a sound signal that will be the track of the air pressure in time. In a formal way a signal is a function in time and can be noted as

$$x(t) \tag{1}$$

When a signal is **processed** we mean it has been changed due its passing throught a **System**, for example if we consider a sound wave, the effect of dampening produced by its passing throught a physical medium as water, can be seen as a natural processing by the *system* (water) to the *signal* (sound). We can note that as

$$y(t) = T[x(t)] \tag{2}$$

where *y(t)* is the signal after it has been processed and *T[]* is the operator which defines the system

When we are relating signals with computers we cannot avoid to mention what is **Sampling**, in brief if we want to memorize a real signal we will be acquiring its value in time intervals, due to hardware bounds no matter what it is impossible to acquire a time continue signal with infinite precision; this procedure is called **sampling** and the signal which is actually memorized is refered as **sampled signal**, if the sampling period between acquired value is constant we can note :

$$\tilde{x}[n] = x(t = n \cdot T_s), \qquad n \in Z \tag{3}$$

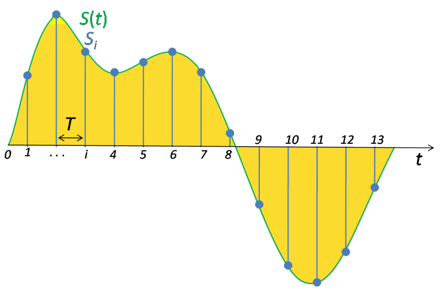


Figure 1 : Signal Sampling

and so a real signal can be seen as the limit of sampling with infinite precision

$$x(t) = \lim_{T_s \to 0} \tilde{x}[n \cdot T_s] \tag{4}$$

So as we can see in *Figure 1* a digital signal is actually stored as a series of unit pulses separated by a time period. That is important as we approach the topic of *signal processing.*

We have already shown in (2) how a processing system can be written as an operator *T[ ].* Speaking of **Linearity** and **Time Invariance**, we say a System is:

1. **Linear** if the response of a linear combination of signals is also the linear combination of the responses of the single signals

$$T[a \cdot x_1(t) + b \cdot x_2(t)] + ... = a \cdot T[x_1(t)] + b \cdot T[x_2(t)] + ... \tag{5}$$

2. **Time Invariant** if the response does not depend of the time at which the input enters the system

$$T[x(t)] = y(t) \Rightarrow T[x(t + t_0)] = y(t + t_0) \tag{6}$$

if we consider the entire sampled signal as the superposition of discrete pulses, if we call $\delta[n]$ the zero-centered unit pulse the sampled signal will be

$$\tilde{x}[n] = \sum_{k=-\infty}^{\infty} \tilde{x}_k \cdot \delta[n-k] \tag{7}$$

that is a key concept for signal processing because if we consider a Linear Time Invariant System we can applies such properties and (7) to get

$$y[n] = T[x[n]] = T[\sum_{k=-\infty}^{\infty} \tilde{x}_k \cdot \delta[n-k]] = \sum_{k=-\infty}^{\infty} \tilde{x}_k \cdot T[\delta[n-k]] \tag{8}$$

in other words the response of a Linear Time invariant System to a whatever signal $x[n]$ can be obtained by knowing its response to the unit pulse signal T[$\delta[n]$] and appling the formula (8). From now on the unit pulse response will be refered as 'impulsive response'

$$T[\delta[n]] = h[n] \tag{9}$$

and the formula (8) is the so called 'discrete time convolution' and is often refered as

$$\sum_{k=-\infty}^{\infty} x[k] \cdot h[n-k] = x[n] * h[n] \tag{10}$$

2.2 Frequency Domain

When we work with signal it's useful to keep in mind the duality that exist between a in time domain representation and a frequency based one. The key relation which links this two approach is the Fourier Transform Relation; given a time based signal $x(t)$ we can evaluate the relative frequency based one $X(f)$ by computing :

$$X(f) = \int_{-\infty}^{\infty} x(t) \cdot e^{-j2\pi ft} \tag{11}$$

that is also valid for discrete time signals as

$$X(F) = \sum_{k=-\infty}^{\infty} x[k] \cdot e^{-j2\pi kF} \qquad F = f \cdot T_s \tag{12}$$

in the discrete case its worth noting that the Transform $X(F)$ is a periodic function in the normalized frequency F with period 1 , moreover it's easy to show that its relation with the continuous time Transform follows

$$X(F) = F_s \sum_{k=-\infty}^{\infty} X(f - kF_s) \tag{13}$$

Now, if we apply (12) to (10) we get the **response** of a LTI System in Frequency domain in a result which is known as 'Convolution Thorem'

$$Y(F) = \sum_{k=-\infty}^{\infty} y[k] \cdot e^{-j2\pi kF} = ... = X(F) \cdot H(F) \tag{14}$$

So the Fourier Transform of the impulsive response *H(F)* can be seen as the gain applied to the frequency F of the input signal, in general $H(F)$ is a complex number meaning that it alters both the magnitude and phase of input's frequencies.

Something we can deduce from (14) is that if a System is LTI it won't be able to generate additional frequencies to the input , meaning that if we pass a pure sine function in an LTI we will get another pure sine as the output varied in amplitude and phase.

2.3 Nonlinear Systems

Once the previous notions on LTI Systems in signal processing has become clear we can advance to the study of Nonlinear one. A NLS violates the property of linearity meaning that (5) is not valid. So we can deduce the system's operator $T[\,]$ applies a sort of Nonlinear function to the input in contrary to what it has been shown in (8) which is a linear combination of shifted version of the input as we can see :

$$y[n] = T[x[n]] = \sum_{k=-\infty}^{\infty} h[k] \cdot x[n-k] \tag{15}$$

Moreover looking at formula(15) we say the system refered by the operator $T[\,]$ is **Dynamic**, meaning that it holds memory so that the output at time istant n depends on input's past or future ($x[n-k]/x[n+k]$); althought if we'd have a combination of just $x[n]$ in the relation the System would be called **Static**, memoryless.

If we now consider a Nonlinear System, Dynamic, Time-invariant, we can imagine to represent it in a 2 parts cascade : a Linear Dynamic System followed by a Nonlinear Static one. We now choose to define the Nonlinear Static as a Taylor Series power relation :

$$y[n] = x[n] + x[n]^2 + x[n]^3 + x[n]^4 + ... \tag{16}$$

That is actually a forcing given the fact that a Nonlinear System could be implemented by appling a whatever nonlinear function ($log(x), tanh(x)$...) However as we'll see for Audio Distortion this choice suits well.

Then the Linear Dynamic part can be easily model as we have previously discussed in (10) :

$$y[n] = x[n] * h[n]$$

So the entire Nonlinear, Dynamic System will be represented in a Graph style like :

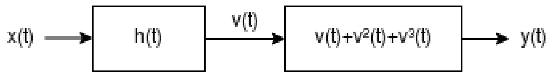


Figure 2: LTI-NL Cascade

if we explicit y it results :

$$y[n] = x[n] * h[n] + (x[n] * h[n])^2 + (x[n] * h[n])^3 + ...$$

by writing the convolutions in explicit form we get :

$$y[n] = x[n] * h[n] + \sum_{k_1=-\infty}^{\infty} \sum_{k_2=-\infty}^{\infty} h[k_1]h[k_2]x[n-k_1]x[n-k_2] + ...$$

we now notice that the higher-order convolutions, in contrary to the first term, are related to multidimensional sums.

if we group the impulse responses as

$$\begin{aligned} h_2[k_1,k_2] &= h[k_1]h[k_2] \\ h_3[k_1,k_2,k_3] &= h[k_1]h[k_2]h[k_3] \\ &... \end{aligned} \tag{17}$$

Those $h_i[k_1,...,k_i]$ are the so called **Volterra Kernels**, multidimensional impulse responses which appears in the general **Volterra Serie** representation for nonlinear System's response :

$$\begin{aligned} y[n] &= \sum_{m=0}^{M} H_m[x[n]] \\ H_m[x[n]] &= \sum_{k_1=-\infty}^{\infty} .. \sum_{k_m=-\infty}^{\infty} h_m[k_1,...k_m]x[n-k_1]...x[n-k_m] \end{aligned} \tag{18}$$

The previous concept of Volterra Serie is crucial for what it will be treated next. In particular it has given to us a way to define a Model for an unknow nonlinear system in a black-box approach, in the next section its theory will be exhamined

*

2.4 Volterra Serie

Please notice that this section has the purpose of presents some theoretical results on the Volterra Serie, however they are not essential to understand the modeling method that will follow in this thesis, so you can fell free to skip that for a faster reading. This notions comes from some lessons I found from Dr Win Van Drogen at Chicago University [1].

2.4.1 Super Imposition

We have seen that a nonlinear system is modelled with the Volterra Serie throught a sum of **Volterra Operators** ($H_n[\ ]$) applied to the input, actually the first-order Volterra Operator $H_1[\ ]$ is the convolution operator that we've seen for LTI systems.

$$H_1[x[n]] = \textstyle\sum_{k=-\infty}^{\infty} h[k]x[n-k]$$

if we try to verify what happens if we give to a Volterra Operator a super imposition of different inputs we get :

$$H_1[x_1[n] + x_2[n]] = \sum_{k=-\infty}^{\infty} h[k](x_1[n] + x_2[n]) = H_1[x_1[n]] + H_1[x_2[n]] \tag{19}$$

as we expect for the Linear Operator Linearity is valid, however for an higher order operator

$$\begin{gathered} H_2[x_1[n] + x_2[n]] = \\ \sum_{k_1=-\infty}^{\infty} \sum_{k_2=-\infty}^{\infty} h_2[k_1, k_2](x_1[n-k_1] + x_2[n-k1])(x_1[n-k_2] + x_2[n-k_2]) \\ ... = H_2[x_1[n]] + H_2[x_2[n]] + 2H_2[x_1[n], x_2[n]] \end{gathered} \tag{20}$$

it has appeared a terms that indicates the deviation from linearity

$$2H_2[x_1[n], x_2[n]] = 2 \sum_{k_1=-\infty}^{\infty} \sum_{k_2=-\infty}^{\infty} h_2[k_1, k_2]x_1[n-k_1]x_2[n-k_2] \tag{21}$$

k_1, k_2 are dummy variables that can be swapped, that's why :

$$H_2[x_1, x_2] + H_2[x_2, x_1] = 2H_2[x_1, x_2]$$

We can apply a similar and it's easy to show how the Volterra Operators behaves on input scaling

$$\begin{aligned} H_1[Cx[n]] = CH_1[x[n]] \\ H_2[Cx[n]] = C^2H_2[x[n]] \end{aligned} \tag{22}$$

2.4.2 Kernel Sampling

We now show how its easy to use the previous concept to identify the Higher Orders Volterra Kernels, here it will be show for a 2nd order System, however you could theoretically generalize it for higher orders. Giving a pair of time shifted pulses as input to the 2nd order Volterra Operator it results :

$$\begin{aligned} H_2[\delta[n-\tau_1], \delta[n-\tau_2]] &= \sum_{k_1=-\infty}^{\infty} \sum_{k_2=-\infty}^{\infty} h_2[k_1, k_2]\delta[n-\tau_1-k_1]\delta[n-\tau_2-k_2] \\ &= h_2[n-\tau_1, n-\tau_2] = h_2[\tilde{n}_1, \tilde{n}_2] \end{aligned} \tag{23}$$

the relation between $\tilde{n}_1, \tilde{n}_2, \tau_1, \tau_2$ is :

$$\tilde{n}_1 = \tilde{n}_2 + (\tau_1 - \tau_2)$$

that's a 45° line at quote $(\tau_1 - \tau_2)$ on the 2D plain $\tilde{n}_1/\tilde{n}_2$ which is the Domain of the Volterra kernel $h_2[\tilde{n}_1, \tilde{n}_2]$.
That means that the result of $H_2[\delta[n-\tau_1], \delta[n-\tau_2]]$ will be the diagonal slice of the kernel on that line.
If we use the superposition relation:

$$H_2[\delta_1, \delta_2] = \frac{H_2[\delta_1 + \delta_2] - H_2[\delta_1] - H_2[\delta_2]}{2} \tag{24}$$

assuming we perform a correction in which we subtract the Linear term ($H_1[\,]$) to the response, $H_2[\delta_1 + \delta_2]$ will be the System's response to both the pair of pulses, $H_2[\delta_1], H_2[\delta_2]$ is the response to the pulses in isolation. So with this method we could sample every of the 2D kernel slices, by varing the time delays of the pulses to be sent to the NLS.

It's useful to keep in mind the **Symmetry** property of the Volterra Kernels, in this case we can notice how

$$h_2[\tilde{n}_1, \tilde{n}_2] = h_2[\tilde{n}_2, \tilde{n}_1]$$

as the order of the pulses is not relevant, the in-between delay $(\tau_1 - \tau_2)$ matters.

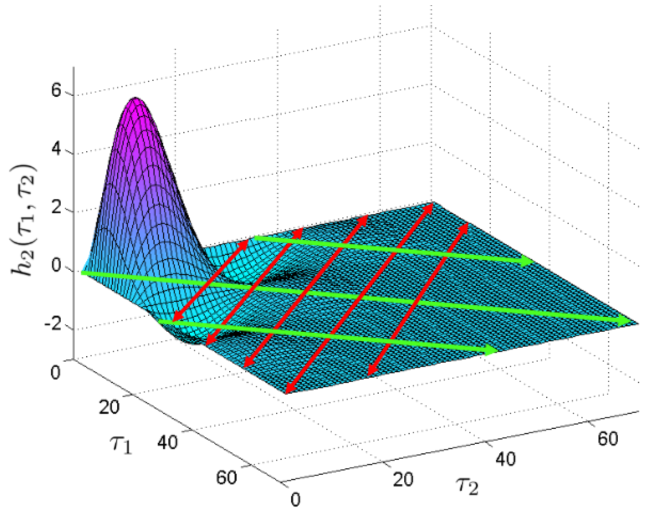


Figure 3 : A 2nd order Volterra Kernel

2.4.3 Harmonic Generation

An interesting behaviour of higher order Volterra operators can be observed when we give in as input a pure tone. As we'll see the harmonics till the nth will be generated over that pure tone, let's demonstrate it for order 2. We have

$$y[n] = H_2[Acos(\omega n)] = H_2[\frac{A}{2}(e^{j\omega n} + e^{-j\omega n})] \tag{25}$$

now we apply the formula from (20)

$$y[n] = H_2[\tfrac{A}{2}e^{j\omega n}] + H_2[\tfrac{A}{2}e^{-j\omega n}] + 2H_2[\tfrac{A}{2}e^{j\omega n}, \tfrac{A}{2}e^{-j\omega n}]]$$

we can calculate those terms

$$\begin{aligned} H_2[\frac{A}{2}e^{j\omega n}] &= \sum_{k_1=-\infty}^{\infty}\sum_{k_2=-\infty}^{\infty} h_2[k_1,k_2]\frac{A^2}{4}e^{j\omega(n-k_1)}e^{j\omega(n-k_2)} = \\ = \frac{e^{2j\omega n}A^2}{4} &\sum_{k_1=-\infty}^{\infty}\sum_{k_2=-\infty}^{\infty} h_2[k_1,k_2]e^{-j\omega k_1}e^{-j\omega k_2} = \frac{e^{2j\omega n}A^2}{4}\Psi(-j\omega,-j\omega) \end{aligned} \tag{26}$$

the function $\Psi(\ )$ has been introduced to ease the formulas writing, we now go on by writing the whole expression of the output :

$$\begin{aligned} y[n] = \frac{A^2}{4}&[e^{j2\omega n}\Psi(-j\omega,-j\omega) + e^{-j2\omega n}\Psi(j\omega,j\omega) + \Psi(j\omega,-j\omega) + \Psi(-j\omega,j\omega)] \\ &= \frac{A^2}{4}[e^{j2\omega n}\Psi(-j\omega,-j\omega) + (e^{j2\omega n}\Psi(-j\omega,-j\omega))^* + \Psi_1 + \Psi_2] \\ &= \frac{A^2}{4}[2Re\{e^{j2\omega n}\Psi(-j\omega,-j\omega)\} + \Psi_1 + \Psi_2] \\ &= \frac{A^2}{4}[2\Psi(-j\omega,-j\omega)cos(2\omega n) + \Psi_1 + \Psi_2] \end{aligned} \tag{27}$$

we can say that $\Psi(\ )$ terms will be number coefficents and we can see how it has appeared a cosine term at a frequency double the input's one.
Furthermore we could demonstrate that also for for higher orders, resulting that the nth Volterra Operator will respond to a pure tone with its nth harmonic.

*

2.5 Hammerstein Serie

In the previous section we have analyzed the theory beyond the Volterra Serie and one of the most important result we have deduced is that the nth-order Volterra Operator responds to a pure tone with its nth harmonic. That is relevant for audio distorion systems as the major goal for such devices is to generate harmonics. However that's also where we have the Volterra Serie's limits; if we imagine to propose a model that behaves in a similar way to an analog nonlinear electronic circuit, then the number of harmonics to be generated should be high(theoretically infinite) and so the model's order should be the high as well. However the memory and processing cost grows exponentially with the order and so an high order Volterra Model is not only too large to be stored but also unfaisible for a real time application.

That's why in audio applications the Volterra Model needs to be eased and a particular class of Volterra Serie has been proposed.

The **Hammerstein Serie** is a Volterra Serie in which the Volterra Kernels are diagonal, meaning that

$$h_n[k_1, k_2, ..., k_n] \neq 0 \Leftrightarrow k_1 = k_2 = ... = k_n = \mathbf{k}$$

and the Serie becomes :

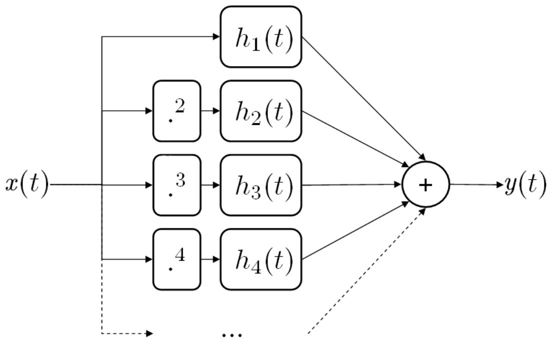
$$y[n] = \sum_{k=-\infty}^{\infty} h_1[k]x[n-k] + \sum_{k=-\infty}^{\infty} h_2[k]x[n-k]^2 + ... + \sum_{k=-\infty}^{\infty} h_M[k]x[n-k]^M$$

which can be easly expressed with the convolution operator as

$$y[n] = \sum_{m=1}^{M} h_m[n] * x[n]^m \tag{28}$$

3 Modeling

Considering the theoretical background that has been referenced it turns out that a black box Modeling method for an audio Nonlinear processing System turns out well to be the implementation of the Hammerstein Serie. To implement an M order Hammerstein Serie according to eq (28) we need to sample from the system the relates Hammerstein Kernels $h_n[\,]$. Each of them will later filter the signal raised to the relative power , then the results is summed in parallel according to the following scheme :



we will now proceed explaining a method to sample the Kernels based on a particular signal called 'Exponential Sine Sweep", I'll try to give the reader an understanding of the validity and limits of this method, that it's presented on [2],[3],[4].

3.1 Hammerstein Kernel and HHFR

Basically we want a way to figure out the Hammerstein's Kernels from an unknow nonlinear system, we kwow that according to the Hammerstein's Model that its response follows :

$$y[n] = \sum_{m=1}^{M} h_m[n] * x[n]^m \tag{29}$$

We won't get those kernels $h_m[n]$ directly, let's consider a new model for the system :

$$y[n] = \sum_{m=1}^{M} h'_m[n] * x[m \cdot n] \tag{30}$$

the model in (30) isn't an Hammerstein neither a Volterra one, however it's interesting if compared to (29) when we have a sinusoid as input.
If $x[n] = sin[\omega \cdot n]$ the model (30) becomes

$$y[n] = \textstyle\sum_{m=1}^{M} h'_m[n] * sin[m \cdot \omega \cdot n]$$

We see how the system's response is modeled as a sum of linear filtering of the inputs harmonics $sin(m \cdot \omega \cdot n)$. That's why the kernels $h'_m[n]$ are usually refered as **Higher Harmonics Frequency Responses**(HHFR).

Now let's see how we can relate this two models, it's known from trigonometry that the nth power of a sine(cosine) function can be expressed as a linear combination of its harmonics to the n :

$$sin(x)^n \propto \sum_{k=0}^{n} a_k sin(k \cdot x) \tag{31}$$

That means it 's theoretically possible if we relate the models in (29),(30) with $sin(x)$ as input, to express the sin power terms $sin(\omega x)^n$ in (29) as a combination of the harmonics $sin(m \cdot \omega x)$. By doing that we can now figure out the dependency of the Hammerstein kernels ($h_m[n]$) with the HHFR ($h'_m[n]$), however that would be examined later in details.

3.2 Exponential Sine Sweep Method

We have seen how the Hammerstein Kernels can be derived from the HHFR, that is important because it exists a method to easily get the HHFRs from a Nonlinear System, let's see it.
The HHFRs were introduced as a way to model a NLS by:

$$y[n] = \textstyle\sum_{m=1}^{M} h'_m[n] * x[m \cdot n]$$

however this model can't be easily deconvolved because it's actually a parallel of various LTIs, actually it would be preferable to have a single convolution relation, for example we could imagine to have a single array containing the sequence of time shifted HHFRs , formally

$$y[n] = (\sum_{m=1}^{M} h'_m[n + \Delta_m]) * x[n] \tag{32}$$

now we do have a single convolution to be deconvolved. That is the question : does it exists a certain input $x[n]$ such as this relation is valid ?

$$\sum_{m=1}^{M} h'_m[n] * x[m \cdot n] = (\sum_{m=1}^{M} h'_m[n + \Delta_m]) * x[n] \tag{33}$$

if we perform an index substitution $n' = n + \Delta_m$ we see that the input $x[n]$ should respect the property

$$x[m \cdot n] = x[n + \Delta_m] \tag{34}$$

an interesting function family that follows that is the **Exponential Sine Sweep**

$$x[n] = sin(Ae^{\Phi[n]}) \Rightarrow sin(Ae^{\Phi[m \cdot n]}) = sin(Ae^{\Phi[n]} \cdot e^{\Delta_m = ln(m)}) \tag{35}$$

in a time instant it's a trigonometric function, so that's a perfect candidate to be the measuring function to deconvolve the HHFRs of the system.
The relation becomes

$$y[n] = (\sum_{m=1}^{M} h'_m[n + \Delta_m]) * sin(Ae^{\Phi[n]}) = \sum_{m=1}^{M} h'_m[n] * sin(Ae^{m\Phi[n]}) \tag{36}$$

and if we deconvolve with this kind of input we will get the time ordered HHFRs

$$(\sum_{m=1}^{M} h'_m[n+\Delta_m]) = y[n] *^{-1} sin(\Phi[n]) \tag{37}$$

3.3 Synchronized Sine Sweep Signal

let's see how to define the signal, if $\Phi[n] = Ae^{\alpha \cdot n}$

$$x[n] = sin(A \cdot e^{\alpha \cdot n}) \Rightarrow Ae^{\alpha(n+\Delta_m)} = e^{\alpha \Delta_m} Ae^{\alpha n} \rightarrow \Delta_m = ln(m)/\alpha \tag{38}$$

Now we want a sweep in a certain frequency range $[f1, f2]$ to be done in a time period T, the instantaneous frequency is the derivate of the phase

$$f(t) = \frac{d\{A \cdot e^{\alpha \cdot t}\}}{2\pi \cdot dt}, \qquad f(0) = f_1, f(T) = f_2 \tag{39}$$

so by performing the derivate we get

$$f_1 = \frac{A\alpha}{2\pi} \Rightarrow \alpha = \frac{2\pi f_1}{A} \tag{40}$$

now we explicit f_2 with $\alpha = \frac{2\pi f_1}{A}$

$$f_2 = \frac{A2\pi}{A2\pi} f_1 e^{\frac{2\pi f_1 \cdot T}{A}} \Rightarrow A = \frac{2\pi T f_1}{ln(f2/f1)} \tag{41}$$

and now we can explicit α

$$\alpha = \frac{ln(f_2/f_1)}{T} \tag{42}$$

So we have the explicit formula for the SSS signal from the parameters f_1, f_2, T

$$x[n] = sin(\frac{2\pi T f_1}{ln(f2/f1)} \cdot e^{\frac{ln(f_2/f_1)}{T} \cdot n}) \tag{43}$$

We can now think of **Synchronizing** the signal, meaning that it start at 0 and has a 0 every Δ_m so

$$sin(\Phi[0]) = 0 \Rightarrow 2\pi k = \frac{\tilde{T} f_1}{ln(f_2/f_1)} \cdot 1$$
$$k = \lfloor \frac{\tilde{T} f_1}{ln(f_2/f_1)} \rfloor$$

where $\tilde{T}$ is an estimation of the effective T parameter, than once we set k as an integer we get the effective Time period to use to have a Synchronized signal

$$T_{eff} = \frac{k \cdot ln(f_2/f_1)}{f_1} \tag{44}$$

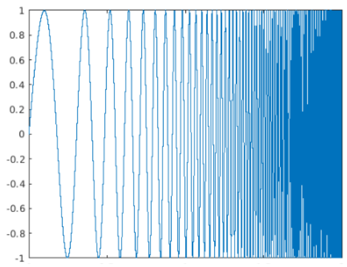


Figure 4 : Synchronized Sine Sweep Signal

3.4 Hammerstein Kernels extraction

by choosing a sinusoidal input for the system we have the following relation between the HHFRs ($h'[n]$) and the Hammerstein Kernels ($h[n]$)

$$\textstyle\sum_{m=1}^{M} h_m[n] * sin[n]^m = \sum_{m=1}^{M} h'_m[n] * sin[m \cdot n]$$

It's known in trigonometry that a power of the sine function can be expressed as sum of its harmonics like

$$\left\{ \begin{array}{l} sin^{2n+1}(x) = \frac{-1^n}{4^n} \sum_{k=0}^{n} (-1)^k \binom{2n+1}{k} sin([2n+1-2k]x) \qquad \forall n \in N \\ \\ sin^{2n}(x) = \frac{-1^n}{2^{2n-1}} \sum_{k=0}^{n-1} (-1)^k \binom{2n}{k} [cos([2n-2k]x) + \frac{1}{2^{2n}}] \qquad \forall n \in N \end{array} \right. \tag{45}$$

That is also valid for the Synchronized Sine Sweep signal as it is a sine in a time instant so if we note

$$ss[n] = sin[\tfrac{2\pi T f_1}{ln(f2/f1)} \cdot e^{\frac{ln(f_2/f_1)}{T} \cdot n}]$$

the modulation property for the SSS signal says

$$ss[[2n+1-2k]n] = ss[n + \Delta_{2n+1-2k}]$$

if we transform in frequency domain we apply the shifting property

$$\mathscr{F}\{ss[n + \Delta_{2n+1-2k}]\} = SS(\omega)e^{i\omega\Delta_{2n+1-2k}}$$

The relation becomes

$$\left\{ \begin{array}{c} SS^{2n+1}(\omega) = SS(\omega)\frac{-1^n}{4^n} \sum_{k=0}^{n} (-1)^k \binom{2n+1}{k} e^{i\omega\Delta_{2n+1-2k}} \qquad \forall n \in N \\ \\ SS^{2n}(\omega) = i \cdot SS(\omega)\frac{-1^n}{2^{2n-1}} \sum_{k=0}^{n-1} (-1)^k \binom{2n}{k} e^{i\omega\Delta_{2n-2k}} \forall n \in N \end{array} \right. \tag{46}$$

in compact form

$$\left\{ \begin{array}{c} SS^{2n+1}(\omega) = a_n \cdot SS(\omega) \qquad \forall n \in N \\ \\ SS^{2n}(\omega) = b_n \cdot SS(\omega) \qquad \forall n \in N \end{array} \right. \tag{47}$$

we can apply that to the HHFR-Hammerstein Kernels relation having the SSS as the input signal

$$\sum_{m=1}^{M} H_m(\omega) \cdot SS(\omega)^m = \sum_{m=1}^{M} H'_m(\omega) \cdot SS(\omega)e^{i\omega\Delta_m} \tag{48}$$

So we can express this system in terms of combination of the $SS(\omega)$ signal, that so can be joined in both sides ans semplified , resulting in a linear system with $H_m(\omega), H_m(\omega)'$ as variables; that means we can finally express the Hammerstein Kernels in function of the HHFRs, in a matrix like notation we have

$$\begin{bmatrix} H'_1(\omega) \\ \cdot \\ \cdot \\ \cdot \\ H'_M(\omega) \end{bmatrix} = T \begin{bmatrix} H_1(\omega) \\ \cdot \\ \cdot \\ \cdot \\ H_M(\omega) \end{bmatrix} \tag{49}$$

where the matrix T can be calcuated from the previous relations, its

$$T_{u,v} = \left\{ \begin{array}{c} \frac{-1^{\frac{1-u}{2}}}{2^{v-1}} \binom{v}{\frac{v-u}{2}} \qquad \forall v >= u \ mod(\frac{u+v}{2}) == 0 \\ \\ 0 \qquad else \end{array} \right. \tag{50}$$

3.5 Modeling Method Overview

The previous section's results can be used to develop a method to implement an Hammerstein model which mimics the behaviour of a NLS, we define the necessary steps :

1. Generate the Exponential Sine Sweep which sweeps in decent amount of time in the frequency range we are interested in.
2. Give to the Nonlinear System the Sine Sweep signal as input.
3. Deconvolve the system's response to the Sine Sweep to get the succession of the Higher Harmonics Frequency Responses.
4. Get the diagonal Volterra Kernel from the HHFRs by multiplication in Frequency domain with the Transform matrix T.
5. Implement the Hammerstein Model as a parallel of LTI filtering of the power raised input and the relative Kernel

3.6 Appendix: deconvolution stage

We have seen how if the NLS system has the SS signal as input is valid

$$y[n] = \sum_{m=1}^{M} h'_m[n] * ss[m \cdot n] = (\sum_{m=1}^{M} h'_m[n + \Delta_m]) * ss[n]$$

and so we can get the succession of the HHFRs $h'[n]$ if we deconvolve the output

$$\sum_{m=1}^{M} h'_m[n + \Delta_m] = y[n] *^{-1} ss[n]$$

the most intuitive way to deconvolve is by using the direct and inverse Fourier Transform (FFT algorithm in discrete time):

$$\sum_{m=1}^{M} h'_m[n + \Delta_m] = \mathscr{F}^{-1}\{\frac{\mathscr{F}\{y[n]\}}{\mathscr{F}\{ss[n]\}}\}$$

However in this way the HHFRs spectrum would be band limited in frequency as the SS signal's spectrum is limited $[f_1, f_2]$ in the frequency sweep's range, that results in a cut in the HHFRs of high order. A more accurate way to perform the deconvolution while mantaining the full frequency range for the HHFRs is by the inverse filtering approach. In substance we say the inverse filter of a signal $x[n]$ is

$$\tilde{x}[n] \qquad t.c. \qquad x[n] * \tilde{x}[n] = \delta[n]$$

So if we convolve a signal with its inverse filter we get a pulse, that is interesting if we consider that its easy to proof for a LTI system the following

$$y[n] = h[n] * x[n] \;\Rightarrow\; h[n] = y[n] * \tilde{x}[n]$$

so we can extract the impulse response from an LTI System by convolving the output $y[n]$ with the input's inverse filter $\tilde{x}[n]$.
That been said in the case of the Sine Sweep signal its inverse filter is, in frequency domain :

$$\tilde{SS}(f) = 2\sqrt{f/L \cdot exp\{-i2\pi(L \cdot f(1 - log(f/f1)) - f1) - 1/8)\}}$$

and so we can obtain the HHFRs by deconvolution if we do

$$\sum_{m=1}^{M} h'_m[n + \Delta_m] = \mathscr{F}^{-1}\{\mathscr{F}\{y[n]\} \cdot \tilde{SS}(f)\}$$

4 Experiments and Results

I have developped some MATLAB Code to implement the Hammerstein Model as well as the algorithm I have described in section [3.5]. Considering that this Modeling is oriented to audio distortion devices I have chosen a Digital Audio Effect as subject.
In first I have generate on Matlab the SSS signal, then I bounced three different test signals : A pure tone (150Hz) , A clean Electric guitar chord groove and a Trap 808 Bassline. Those are audio signal which differs in quality, the pure tone response displays a clear view of the harmonic generation profile of the System, the 808 Bass character is typically given by the heavy Distortion been applied and so that is also a good candidate to check the quality of the NLS, last but not least the electric guitar is one of the most iconic distorted sound in rock music and that makes, as the 808 bass, a good test subject to evaluate the Model accuracy by ear.

The Audio Effect I have modeled is the iZotope Trash2, in substance that's a virtual distortion devices that can be load in a recording Software (DAW) to process in real time an audio stream. We can take a quick look at its features :

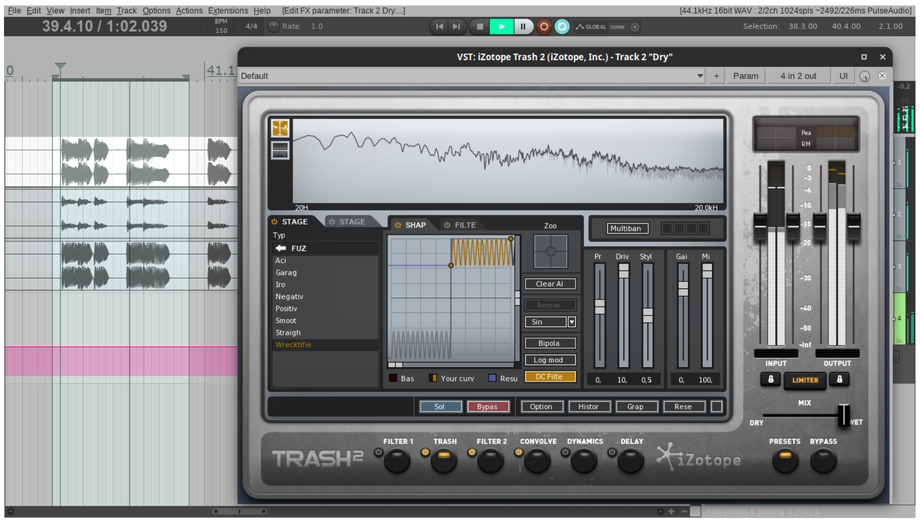


Figure 5 : Trash 2 Waveshaper Stage

As we can see above Trash 2 is composed in a cascade of processing Blocks: The Filter Stage is an Equalizer(Filter Banks), the Trash Stage is a Waveshaping Model, in the middle its displayed the Input/Output relation function which is applied to the signal, here's where we have the major nonlinearities and harmonics generation. The Convolve Stage is a FIR Impulse response's convolution, various IR recorded from Amps,Cabinets and Rooms are avaible.
Different configuration of Trash2's params produces differents distortion behaviours, those configuration can be stored in presets.

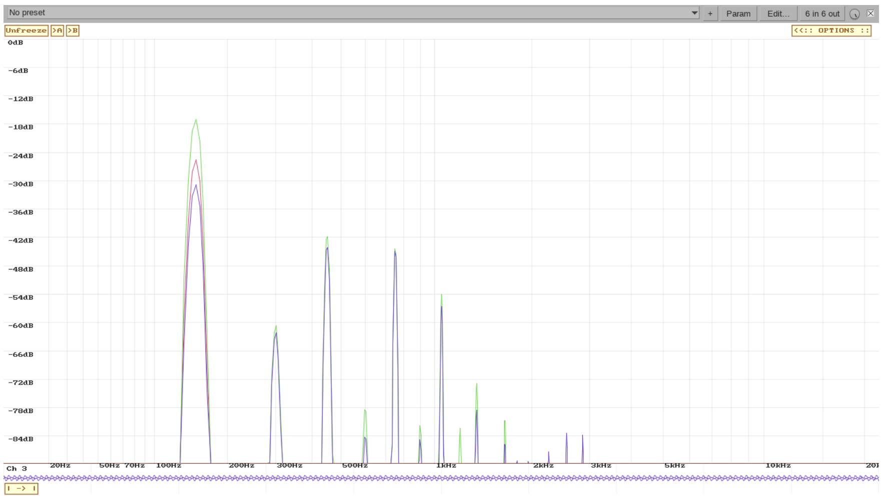
I have applied 3 different presets to the clean guitar sound that are called Broken Amp, Ice Castle and Little Touch and for the bass line the presets applied are called Crusher and Dance Bounce.

4.1 Spectral Results

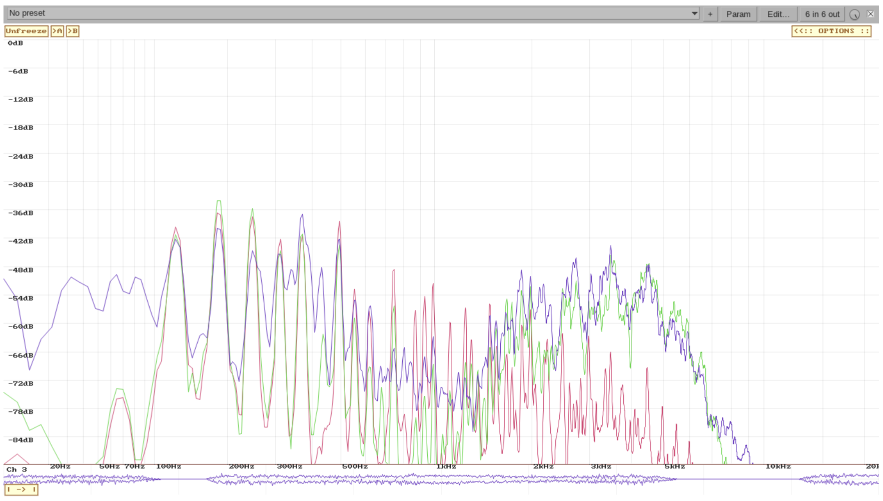
the color map is

1. RED : Non disorted Signal
2. BLUE : Trash2 distorted Signal
3. GREEN : Signal distorted by Hammerstein Modeling

4.1.1 PRESET: Broken Amp, Model Order M=45

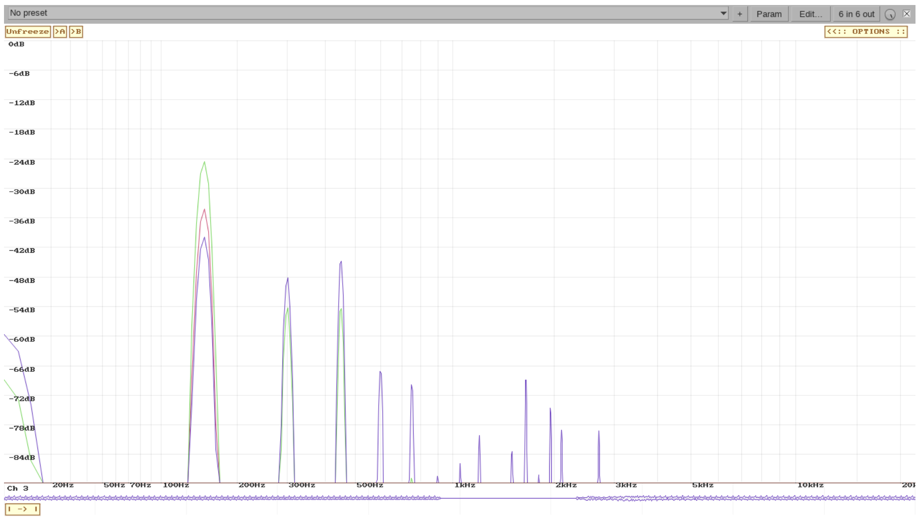


Result 1.1: Sine, Broken Amp M=45

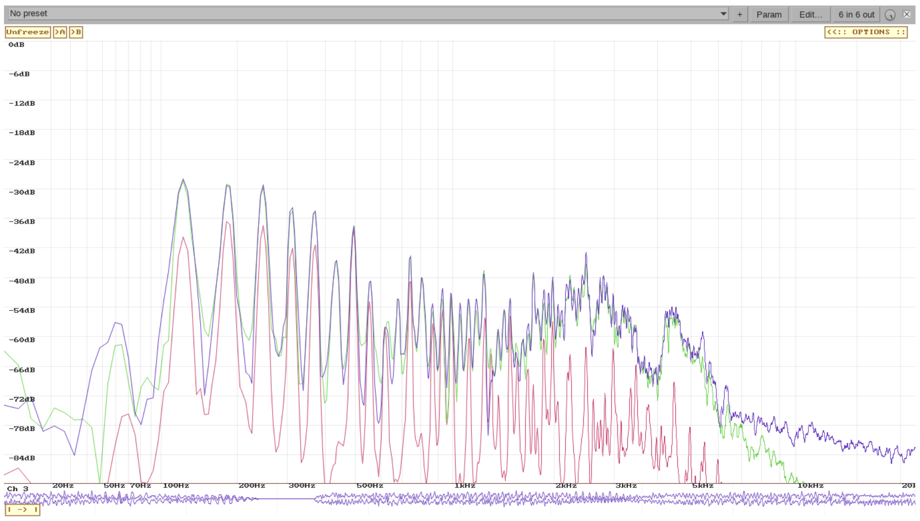


Result 1.2: Guitar, Broken Amp M=45

4.1.2 PRESET: Ice Castle, Model Order M=45

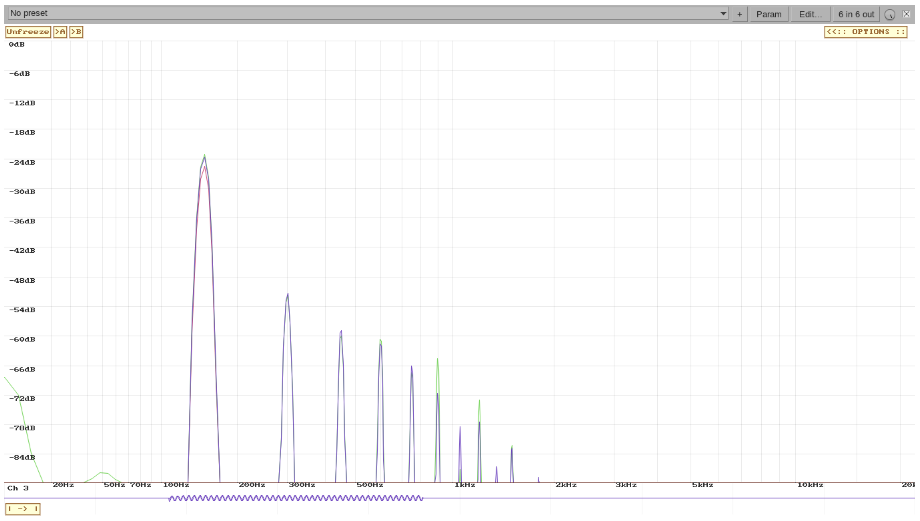


Result 2.1: Sine, Ice Castle M=45

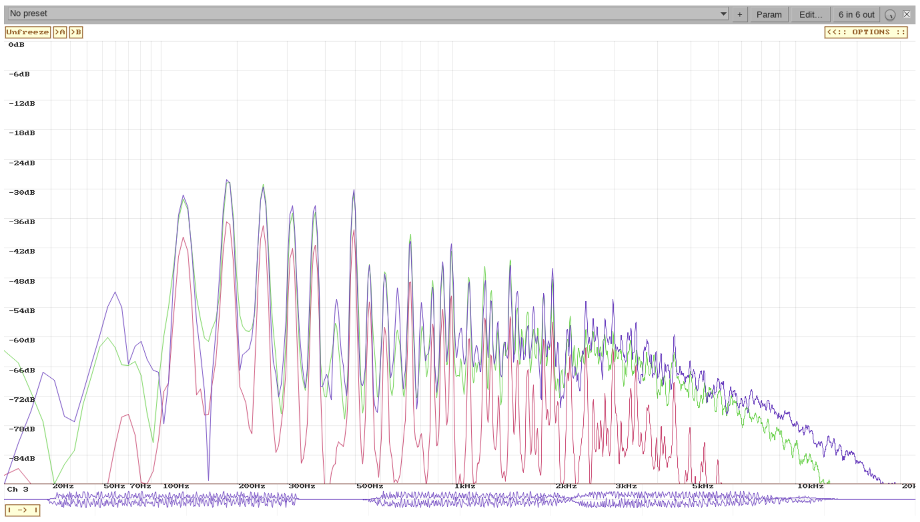


Result 2.2: Guitar, Ice Castle M=45

4.1.3 PRESET: Little Touch, Model Order M=45

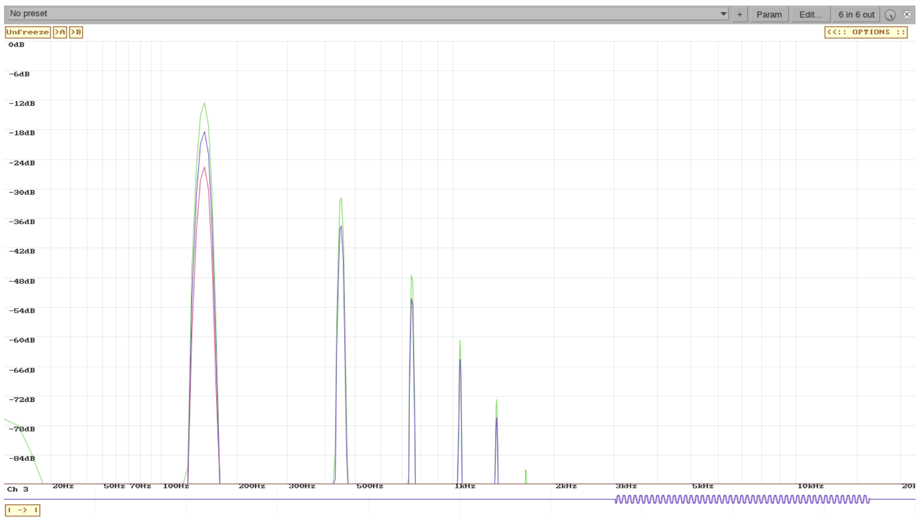


Result 3.1: Sine, Little Touch M=45

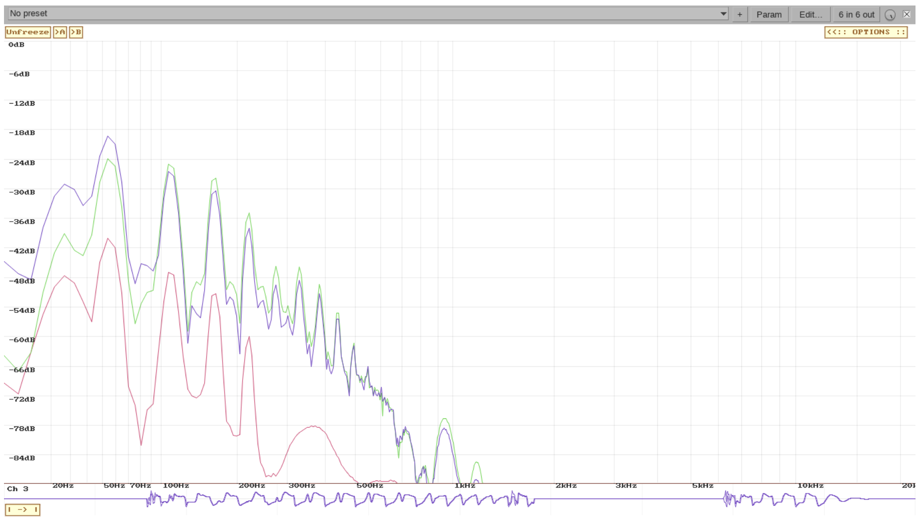


Result 3.2: Guitar, Little Touch M=45

4.1.4 PRESET: Crusher

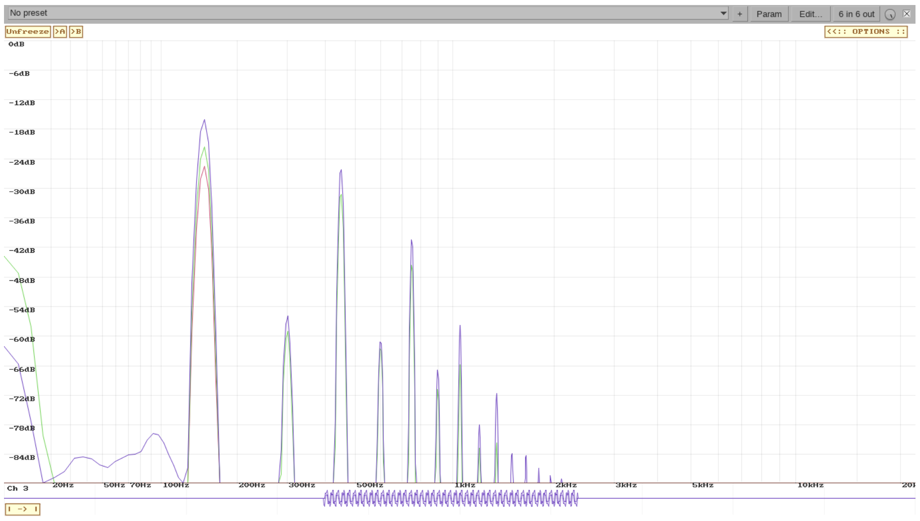


Result 4.1: Sine, Crusher M=45

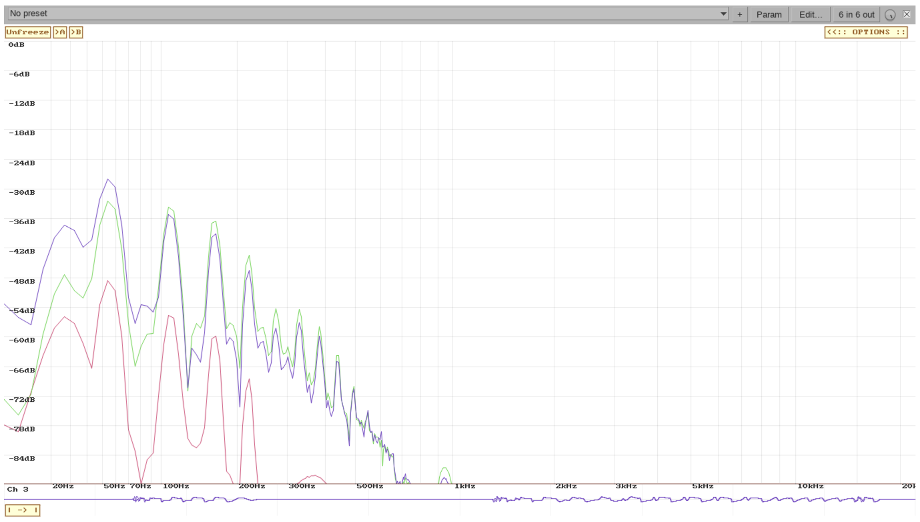


Result 4.2: Bass, Crusher M=45

4.1.5 PRESET: Dance Bounce, Model Order M=45

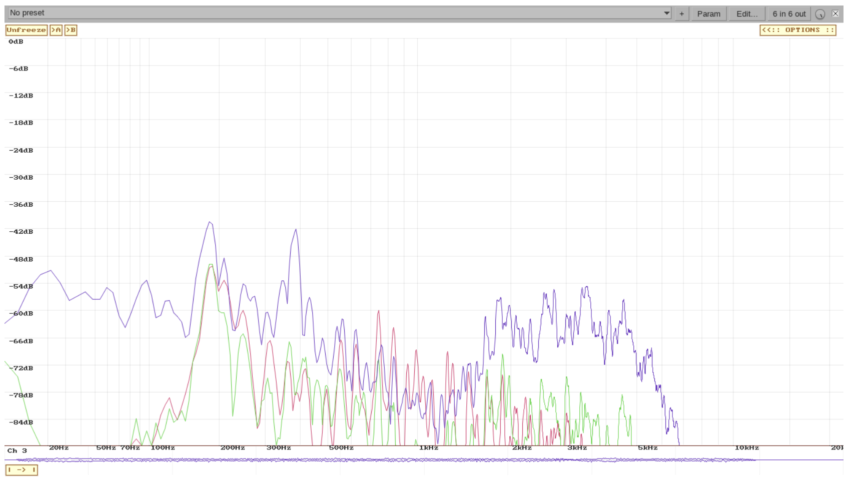


Result 5.1: Sine, Dance Bounce M=45

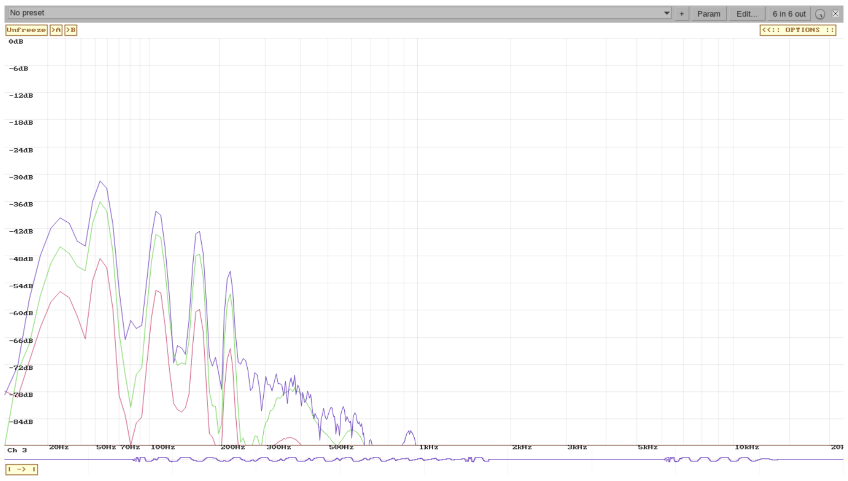


Result 5.2: Bass, Dance Bounce M=45

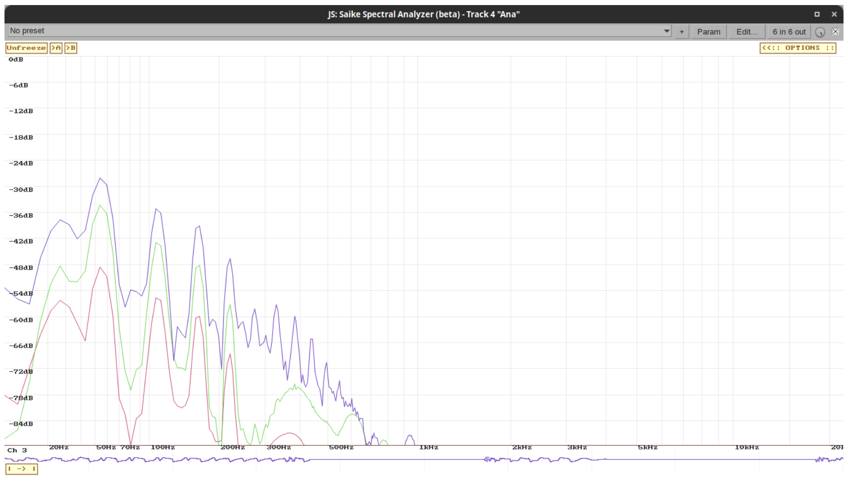
4.1.6 Results for Model Order M=10



Result 6.1: Guitar, Ice Castle M=10



Result 6.2: Bass, Crusher M=10



Result 6.3: Bass, Dance Bounce M=10

4.2 Results analysis

As we can state from the spectral information the Hammerstein Model has resulted in a pretty decent Emulation of the model, that is remarkable if we think that we have followed a pure black box approach. The major deviation in the results is relative to the high frequency response, that is not a surprise as the High frequency behaviour of a system it's typically very sensitive and in Hammerstein Model would take high precision data to extract accurates high order HHFRs/Kernels (I used 16 bit audio data at 192kHz). Moreover something we can ear from the audio results with a pair of good headphones is that the Hammerstein model doesn't take in consideration correlation between the Stereo Channels, we can ear in the preset Ice Castle how the Trash2 plugin affects the stereo image of the input resulting in a very wide sound, that is not correctly modeled by Hammerstein as it just applies the Series to the left and right channel in isolation.

In conclusion we can say that Modeling strategy is interesting for its versatility as it's black box oriented, however has issues in stereo imaging and an accurate high frequencies behaviour needs an high order, that makes this model difficult to implement in a real time enviroment for its cost (we have to perform M LTI-convolutions each having a cost $O(N_x \cdot N_h)$ in relation to the audio buffer's length N_x and the average length of the Kernels N_h).

5 References

1. Signal Processing for Neuroscientists [Wim van Drongelen]
2. Synchronized Swept-Sine: Theory, Application and Implementation [A.Novak, P.Lotton]
3. Hammerstein Kernels Identification by Means of a Sine Sweep Technique Applied to Nonlinear Audio Devices Emulation [T.Schmidtz,JJ.Embrechts]