AALTO UNIVERSITY SCHOOL OF ELECTRICAL ENGINEERING

UNIVERSIDAD POLITÉCNICA DE CARTAGENA
ESCUELA TÉCNICA SUPERIOR DE INGENIERÍA DE TELECOMUNICACIÓN

BACHELOR THESIS

Analysis of Flanging and Phasing Algorithms in Music Technology

Author: Antonio Fuentes Ros
Director: Vesa Välimäki
Co-director: Rafael Toledo-Moreo

October 2019

Acknowledgements:

I would like to thank my parents for making this experience possible and for the constant support and love received from them.

I would also like to express my gratitude and appreciation to Rafael and Vesa, as they have invested hours of their time in helping me leading this project.

Also, many thanks for the support received from the amazing friends I have, as well as the ones I have met throughout this beautiful experience.

INDEX

FIGURE INDEX

1. INTRODUCTION

1.1 Content and structure of the project

As a result of my interest in music, I decided to use all the knowledge acquired along the years studying Telematic Engineering to fully understand how these effects presented in this project work.

This research project consists of a study of digital signal modulation algorithms in filters used in the music field. Specifically, two big group of delay filters were studied: flanger and phaser filters. Flanger and phaser effects are modulation effects. This means the audible signal or carrier is being modified (modulated) by another signal (modulator) while the filter is operating. All the effects presented in this project modify the signal phase in a different way. To begin with, both effects will be presented.

The flanger effect naturally occurs when a noise is heard as the sum of itself in a directly and delayed mode. The most common example of this situation on a day-to-day basis is the sound of an airplane's engine heard directly from the airplane and the reflected sound in a nearby building. As F. A. Bilsen and R. J. Ritsma describe, it was discovered by Christiaan Huygens, a mathematician from the Netherlands, in 1693 at the castle at Chantilly de la Cour in France. Standing in front of a staircase, he realized that the sound from a near fountain was producing a certain pitch, caused by the direct sound of the fountain and the delayed reflections of that sound against the steps of the staircase (Acustica, Vol. 22 pages 64-65).

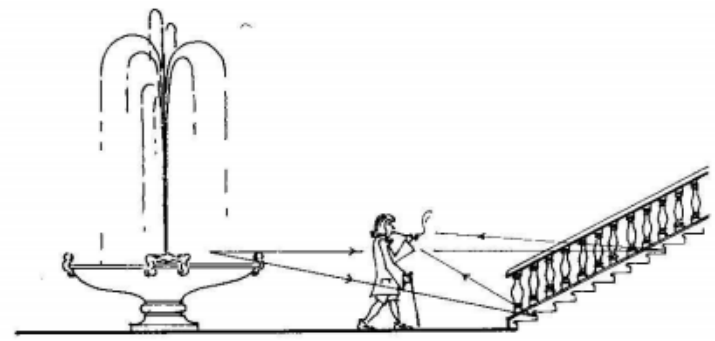

Figure 1. F. A. Bilsen & R. J. Ritsma. "Repetition Pitch and Its Implication for Hearing Theory"[6]

The phaser effect belongs to the linear time-varying audio filters group. Used to "sweeten" the sounds, this one is a popular guitar and keyboard effect used mainly in rock genres in its first years to simulate the flanging effect. In this effect, the signal is split in two parts, modifying one of the parts' phase. When both parts are mixed again, the differences in the phase of each signal cause them to cancel each other at some points. Those points where the signal is decreasing to or increasing from zero are called notches.

This effect is related to the flanger, even used as a synonym, as both effects work sweeping notches in the modified signal, but there is a main difference between them. The number of notches presented on the phasers are limited, and those notches are not uniformly spaced in the spectrum. However, flangers have an infinite series of notches, spaced in a harmonically way. Using the definition of Julius O. Smith III to define both filters, "*we will define a phaser as any linear filter which modulates the frequencies of a set of non-uniformly spaced notches, while a flanger will remain any device which modulates uniformly spaced notches*" (Physical Audio Signal Processing).

Once both groups have been defined, this project will focus on how those effects are going to be studied. Usually, the application of these effects in the Music Technology field is focussed on guitar or synthesizers effects, as those are the most common instruments where filters and effects of this kind are applied. However, nowadays most of the instruments have their electrical version, which facilitate the use of these algorithms.

When studying these effects, they will be studied in three main ways, aiming for a detailed understanding of their behaviour: to begin with, we will see a brief analysis of their transfer functions and the mathematical formula for their modelling, as I believe it is fundamental to understand their behaviour in the frequency spectrum. Then, a presentation of the filter or effect in a digital way in order to understand the different possible implementations of those filters and the components needed for each kind of effect. For the phaser effect, a hardware implementation has been made following the schematics of a given pedal effect in order to be able to study a real implementation of each component of the block diagram.

1.2 Objectives

In this project work, some objectives have been defined to ensure a clear goal and path to follow while working on this research. The overall aim of the project is a better understanding of the studied audio effects in a digital way. The history of the effects is presented, and the implementation of those, both in a mathematical and virtual way are presented. The three main points are:

- Understand the operating principle of each effect in a theoretical level, as well as locate the effect in the history of musical technology.

- Understand the mathematical implementation of the effects at the level of formulas and transfer functions, along with the comprehension of the transfer function of each filter.

- Be able to replicate the effects in a virtual and analogical way, acquiring the knowledge to simulate each effect with electronic components.

2. AUDIO EFFECTS AND ALGORITHMS

2.1 Phaser

2.1.1 Presentation

Based on the variation of the frequency components of a signal, this filter creates a characteristic sound effect by modifying the phase of a signal and adding it to its original form. The process to create a phaser filter consists on splitting the input signal in two parts; one of the parts go directly to an adder while the other part goes through a set of first- or second-order all-pass filters, modifying its phase while maintaining the amplitude. When both parts are added together, the phase difference between them causes the output signal to be a sum or a subtraction of both parts creating peaks and troughs, depending on that phase difference. The points where the output signal is totally zero are the notches of the phaser filter. Due to its effect sound similitudes, it is considered a delay effect, along with the flanger.

2.1.2 Transfer function

Phaser transfer function main point resides in how the all-pass filter modifies the waves in the frequency domain. As this project works with the digital versions of these filter, the transfer function for an analog all-pass filter has to be translated to the digital domain. The formula for the digital approach of a first-order all-pass filter is presented by R. Kiiski, F. Esqueda and V. Välimäki [9] as

$$A(z) = \frac{a_1 + z^{-1}}{1 + a_1 z^{-1}}$$

where the parameter $a_1$determines the break frequency of the all-pass filter. For a stable system, the value of a_1 should be between in the range (-1, 1). In Figure 2 we can see the phase response for an allpass filter for different values of a_1.

Now, to see what happens when a cascade of these first-order allpass filter is implemented, we need to use the Figure 3, obtained from the previous article [9]. There we can see that a cascade of ‘N’ allpass filters creates notches at maximum ‘–Nπ’ rad.,

being ‘N’ a positive even integer. Thus, when adding both parts of the signal, modulated and original, notches will be placed at the frequencies where the phase shift is an exact odd integer multiple of ‘–π’ rad/s or ‘–kπ’, for ‘k’ as an odd number [9].

The problem for this first-order filter implementation resides, as the authors point out [9], in the management of the notches created. For a first-order allpass filter, width and depth cannot be modified independently.

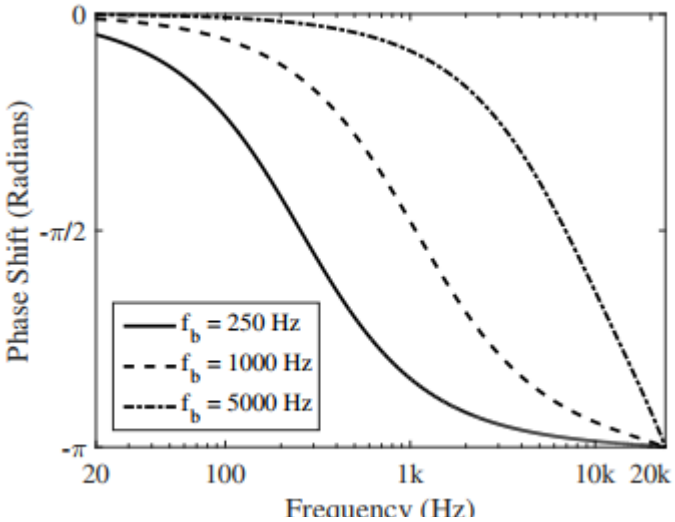

Figure 2. Phase response of a first-order allpass filter with break frequencies 250 Hz (a1 = −0,967), 1000 Hz (a1 = −0,869), and 5000 Hz (a1 = −0,346) [9].

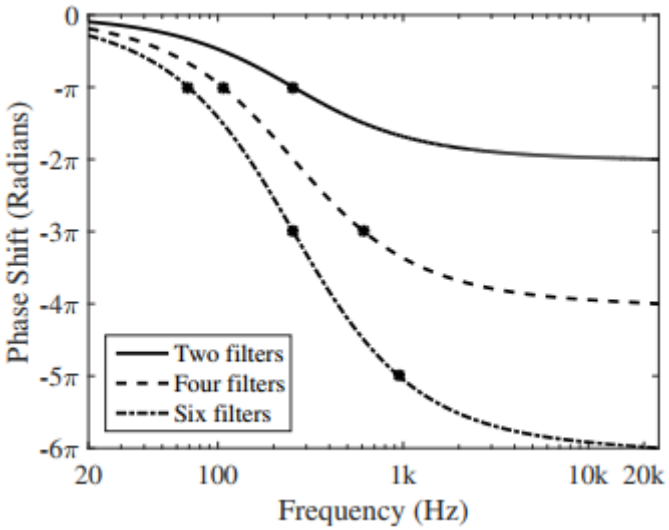

Figure 3. Phase response of two, four, and six cascaded first-order allpass filters with break frequency at 250 Hz (a1 = −0,967). The squares indicate the resulting notch frequencies in each case, when used in a phaser. [9]

In [5], Julius O. Smith presents a possible solution for the notches independent management, residing this solution in the use of second-order allpass filters. The transfer function for the second-order filter is given by

$$H(z) = \frac{a_2 + a_1 z^{-1} + z^{-2}}{1 + a_1 z^{-1} + a_2 z^{-2}}$$

where the values of a_1 and a_2 are:

$$a_1 = -2Rcos(\theta)$$

$$a_2 = R^2$$

with R < 1 and $\theta = \omega_n T \in (0, \pi)$. ω_n is the desired notch frequency, T the sampling interval and R controls the width.

2.1.3 Structure

In order to implement phaser's effect characteristic notches, a special filter type called all-pass is needed. The digital structure of an all-pass filter will be presented later, as it is crucial for the complete understanding of the phaser. Briefly explained, an all-pass filter is a unity-gain filter which allows all frequencies but allow the user to change the phase of the filtered signal. Used to filters passing certain frequencies and attenuating others, this filter may seem odd at first. However, the operation of this filter is as simple as other filters are. Furthermore, unlike most types of filters, using this filter a signal will not suffer any changes on its shape or amplitude. The order of the all-pass filters used in the phaser are normally not higher than 2, and the quantity of all-pass filters is usually between 1 and 10, but this can change.

The key of this filter resides in its ability to change the phase of a signal. Using a black box model and comparing the input signal versus the output one, this filter would only add a delay on the input signal, keeping the input signal's amplitude, due to the phase delay. Being the phase delay introduced by the all-pass filter constantly changing, there will be a 180º delay at certain points, meaning the output signal is just the input signal in its negative form. Adding both parts of the signal at this phase delay will cause a zero on

the output signal, creating the notches of the phaser. In the opposite case, the delay between both parts would be a 360º delay, causing both parts to add increasing the amplitude two times the input signal.

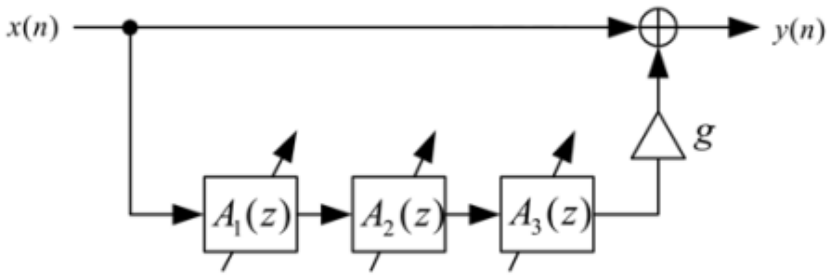

Figure 4. Phaser digital implementation [12]

The schematic represented in Figure 4 is one of the basic phaser implementations. In this case, the phaser has three variable all-pass filters ($A_1(z)$, $A_2(z)$, $A_3(z)$) as well as a gain module *g*.

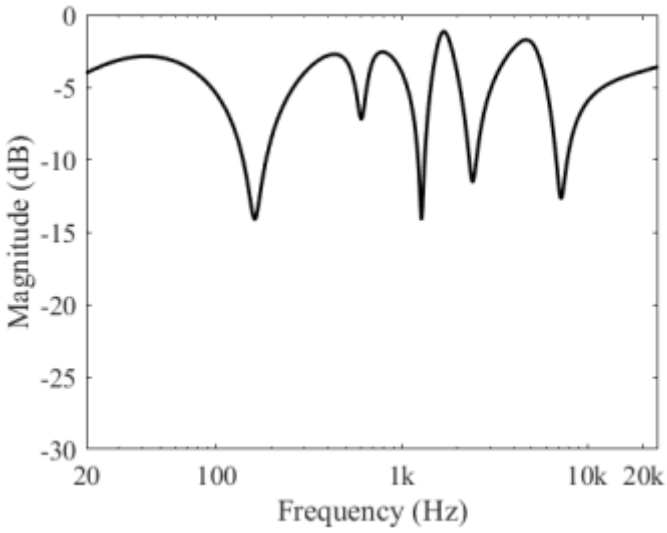

Figure 5. Magnitude response of a phaser having ten first-order allpass filters and a feedback loop. [9]

In the respective spectrogram for the magnitude response presented in Figure 5 we can see how the five notches have their own path on the frequency domain. The zig-zag line described in Figure 6 is drawn by the oscillators controlling each allpass filter's frequency.

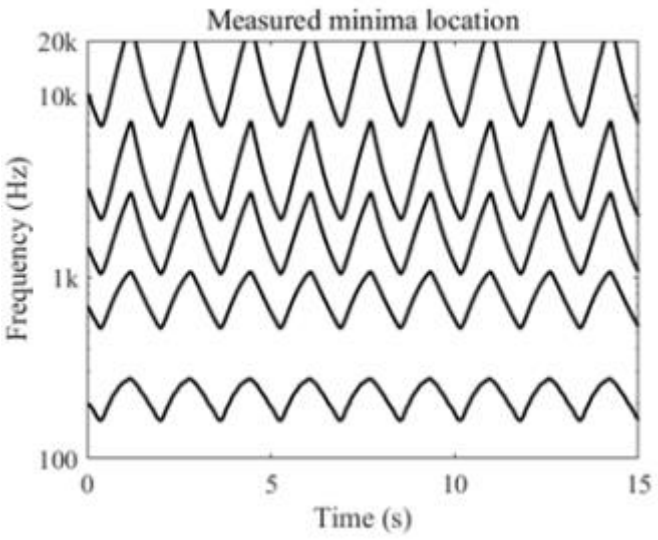

Figure 6. Spectrogram view for the spectrum presented in Figure 3. [9]

The purpose of the gaining module *g* is no other than controlling the amplitude of the filtered part of the signal towards the posterior mixing with the unmodified part. With this gaining block it is possible to modify the depth of the notches.

Other possible implementations of the phaser implies feedback loops with or without delay blocks. These loops act as notch boosters in practice, changing the magnitude response of the usual implementation. However, the most common implementation of the phaser avoids the feedback loop.

2.1.4 Implementation

Aiming for a deeper comprehension of phaser's components and signal transitions along the filter, a real phaser filter has been created following a given schematic of a MXR Phase 90 (Figure 49). The original design of this pedal presents an only potentiometer to control the variation speed of the frequency notches. In the design implemented, another user-accessible potentiometer has been added, controlling the depth of the notches

To start with the phaser implementation, it is necessary to explain and comprehend the structure of the basic component in the phaser: the allpass filter. As commented in

previous sections, a phaser consist of two lines, a clear direct line and a line with a series of allpass filter. There is not a certain quantity of allpass filter for a design, but the average phaser in the market has 2 to 4 allpass filters, creating therefore 1 to 2 notches respectively. As seen in [17], there are 2 ways to implement an allpass filter analogically: using a high-pass filter or using a low-pass filter. On this occasion, as the filter is designed for music purposes, a high-pass voltage controlled implementation will be used:

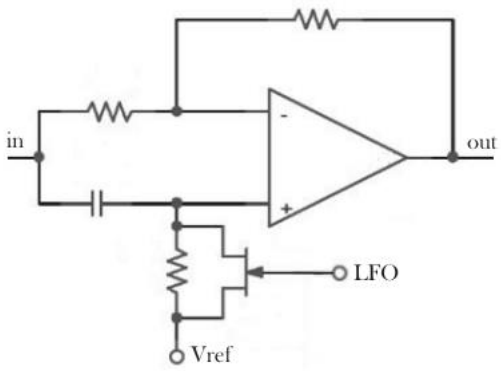

Figure 7. Allpass filter with high-pass implementation.

In Figure 7, the FET implemented in parallel with the resistance is in the ohmic region, creating this way a voltage-controlled phase shifter. The gate voltage is determined by the LFO, as seen in the previous block diagram of the phaser, and it is the responsible for the notch-moving sound.

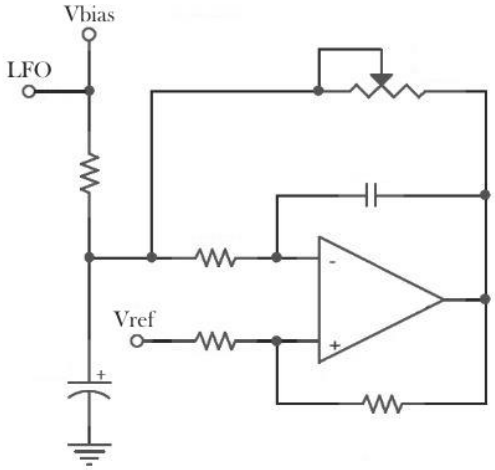

Figure 8. Low frequency oscillator implemented in MXR Phaser 90 design.

Being the LFO the second fundamental component of a phaser, the LFO implemented for this design will be presented, adding an extra control to upgrade the phaser's sound to a more adjustable level. The basic LFO for a MXR Phase 90 flanger is shown in Figure 8. There, it is possible to see how a voltage 'Vref' enters in the circuit and, as seen in [17], a square pulse is generated at the output of the operational amplifier, therefore creating a triangular pulse at 'Vo'. The frequency of these signals can be modified by adjusting the given potentiometer.

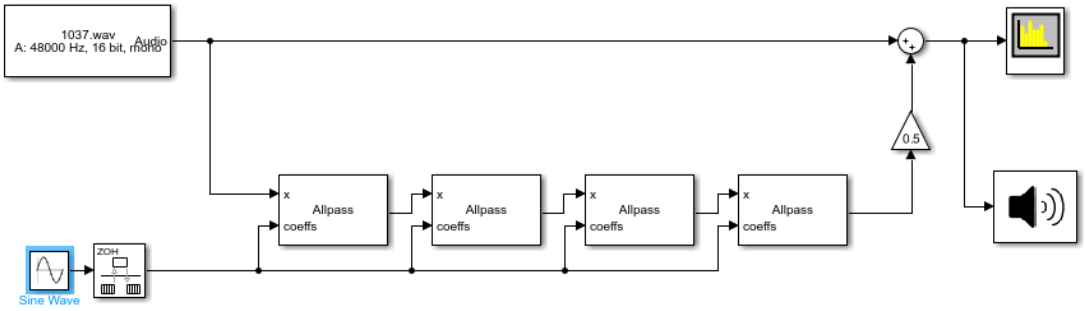

Figure 9. Phaser implemented using Simulink.

The process followed to add an extra control to this implementation starts with the search of the parameter that modifies the depth of the notches, using the implemented phaser on Simulink. Given the spectrograms of the phaser's output signal (figure 6), the "depth" term might lead to a misunderstanding. To avoid the misconception, a brief explanation of the term is necessary to understand the process. When such term is used, it refers to the movement of the notches along the frequency axis.

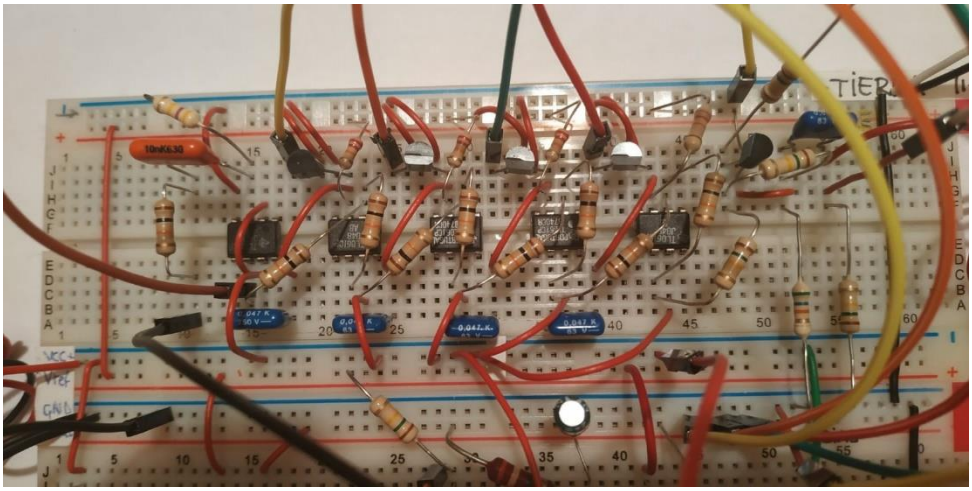

Figure 10. Allpass filter array implemented on a protoboard.

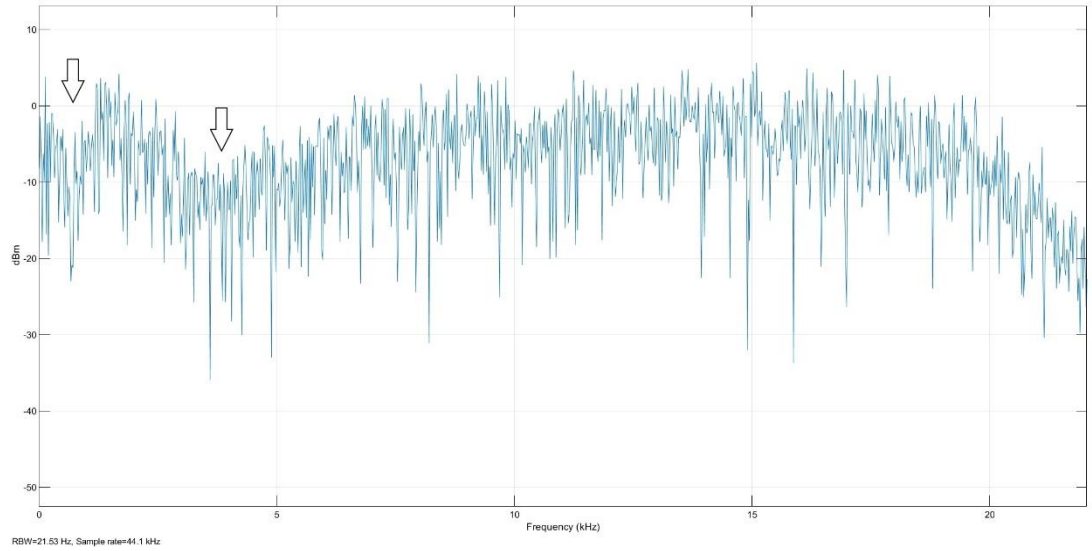

Figure 11. Spectrogram of phaser's output for a pink noise input, notches at their lower frequencies limit.

The image above shows the spectrogram of the output signal for the Simulink phaser implementation for a pink noise input. There, both notches are pointed by the arrows. In that very exact time, notches are at the lowest frequencies they can reach, around 1 and 4 kHz. As time goes by, those notches will move to higher frequencies, until they reach their respective top frequency, as shown in the next picture.

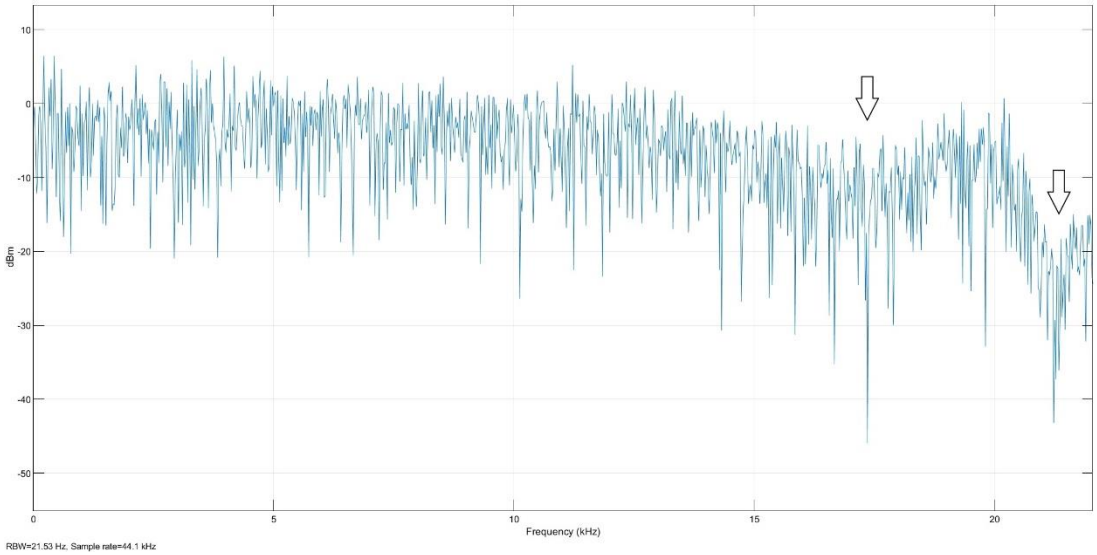

Figure 12. Spectrogram of phaser's output for a pink noise input, notches at their higher frequencies limit.

When adding an extra control to the phaser, the idea was to be able to control the sweep of the notches, regardless the speed of these, reducing the range of movement. Going back to Simulink, it was not hard to find the target parameter to implement. When reducing the LFO amplitude, the range of notches' movement was also reduced, creating a shorter range of movement for the notches.

To achieve this in the built design, it was necessary to find a way to regulate 'Vbias' voltage, therefore regulating LFO amplitude. 'Vbias' and 'Vo' were the tensions used to regulate that voltage, as can be seen below. To operate with those two tensions, a voltage follower was implemented, bringing 'Vo' to the positive (and so to the negative) input of the operational amplifier. Depending on the impedance selected on the 100k potentiometer, 'Vbias' will be affected by 'Vref' and 'Vo', taking a higher or lower amplitude value –with a minimum amplitude given by the lowest 'Vo' value.

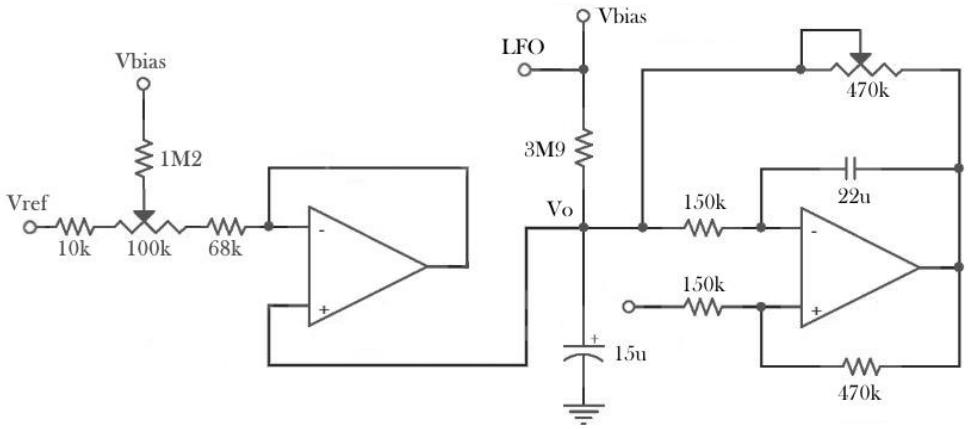

Figure 13. At the left part of the image, the potentiometer added to control the depth of the notches is represented.

Once this control was added, some measures were taken with the oscilloscope, in order to prove its correct operation. Pictures were taken at the time the key points of the notches path: upper and lower limits.

When the lower limit is reached, notches change the direction of its movement, starting the path to the upper limit. At this point, only the right notch can be seen, as the resolution in lower frequencies was not good enough to catch botch notches –there's a difference of less than 1 kHz between them.

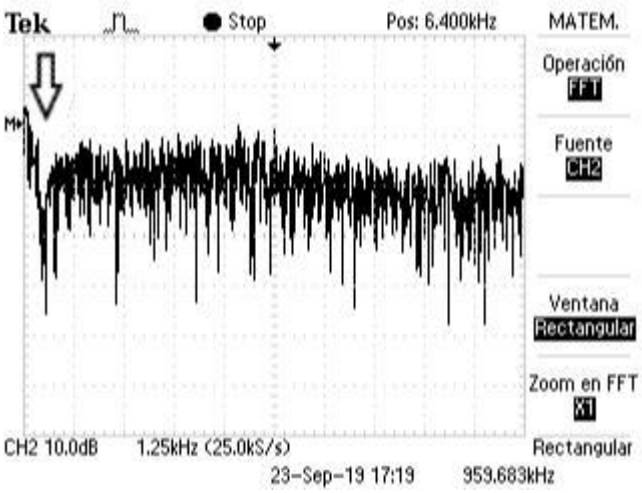

Figure 14. Oscilloscope capture showing the spectrum of the phaser filter implemented. Notches are situated at their low frequency limit.

As time passes, the notch situated at the right will move to higher frequencies, therefore creating more space between both notches and allowing the right notch to appear on the screen. The next picture shows the max range for the notches to move, as the new potentiometer is set with its minimum value.

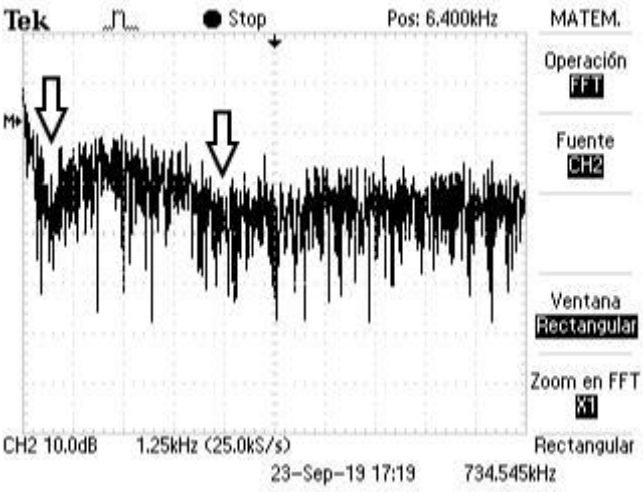

Figure 15. Oscilloscope capture showing the spectrum of the phaser filter implemented. Notches are situated at their high frequency limit while potentiometer is set at its lower range.

When the potentiometer is set at its maximum range, both notches move to higher frequencies. As the space between notches will also increase, the right one will move more kHz compared to the left one, that will barely move.

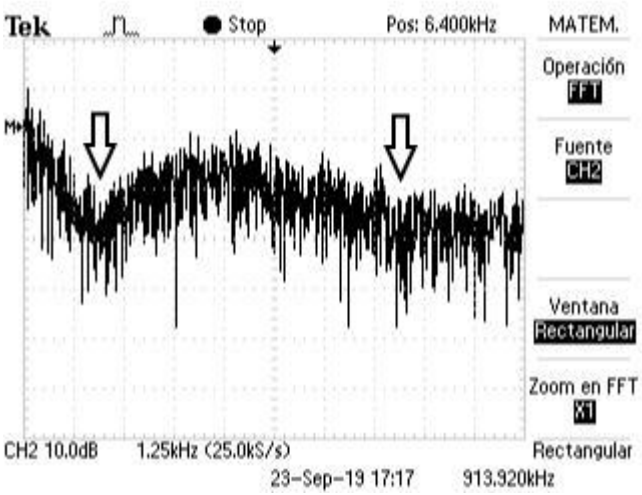

Figure 16. Oscilloscope capture showing the spectrum of the phaser filter implemented. Notches are situated at their high frequency limit while to potentiometer is at its highest range.

Even though it may seem a little change of a few kHz, the difference between scenarios is perfectly audible, giving a sensation of a stronger presence of the phaser when the potentiometer added is at maximum range. To give the implementation a final touch, a double circuit commutator was added. While the ideal assembly for guitar use consist of a two-position double-circuit commutator (there is no need for more states than one going through the effect and another with direct connection between input and output), a triple state commutator was used, as it was not crucial for the right behaviour of the filter and it was nearby during the phaser assembly. One circuit was used to select between the filtered output and the direct connection between input and output. The remaining one was used to implement a bi-colour led, making it easier to identify the state of the circuit: red for a working phaser and green for a direct connection between input and output.

2.2 Flanger

2.2.1 Presentation

Known as the "coloration" of a sound in a reverberant environment, to create a flanging effect, two identical signals have to be mixed together with a light delay on one of them with respect to the other. Applying that delay to one of the tapes while both are running, the signal from the playback head is delayed on time with respect to the other tape, creating this characteristic effect.

$$T = \frac{d_2}{s_2} - \frac{d_1}{s_1}$$

d = distance from record to playback head gaps
s = tape speed

The delay time needed is not defined with precision, but it should be under 10 milliseconds, due to the remarkable difference in the sound when the delay time exceeds that limit. Varying this delay time leads to different names of the flanging effect.

The name of the effect comes from the first methods used to create it: having the same track on two tapes, both tapes were recorded simultaneously while one of the tapes is lightly pressed using the engineer finger in the flange. Alternating this action between both of the tapes, a variable time delay affects both tapes alternatively and so, the flanger effect is created.

From the time domain, the modified signal at the output of the flanger would be similar to the original version, just a little bit delayed on the time axis. But it only takes a simple look to the frequency domain to realize the real effect of the flanger filter. As we will see in the next sections, the spectrum of the output signal changes, showing now the characteristic signature from the flanger: the notches. These notches are different from phaser notches in a reason: they are uniformly distributed in the frequency axis. The process to modify the shape and distribution of these notches will be presented along the next sections, as well as the digital implementation for this filter.

2.2.2 Transfer function

Defining the flanger effect as *a filter which modulates the frequencies of a set of uniformly spaced notches [5]* and regarding what we already know about how this effect works (delay line with a feed-around), we can write the formula of this structure as:

$$y(t) = x(t) + x(t - r(t))$$

y(t) = output signal at time t
x(t) = input signal at time t
t = time (t = 0, 1, 2, 3, …)
r(t) = length of the delay at time t

Being the value of r(t) given by a LFO, the MATLAB implementation created for the formula above looks like this:

```
[x,fs]=audioread('pinknoise.wav');

%_____Controls______
A = 10;
rate = 0.2;
manual = 10;

period = 1/fs;

%_____Y_values______
for t = 1:length(x)-A-manual
    y(t) = (x(t) + x(t +(manual + round(A*sin(2*pi*t*rate*period))))/2);
end

sound(y,44100);
```

being `sin(2*pi*t*rate*period)` the sinusoidal signal used to create r(t). Some controls usually present in flanger implementations were added to this signal, allowing the user to modify different aspects from r(t): with 'A', the amplitude of this signal is modified, giving the user the possibility to control the quantity of notches created. 'Rate', together with 'period', is used to control the frequency of the delaying signal. This way, the flanging effect speed can be modified. 'Manual' variable is used to add an offset to r(t). The effect of this control variable would be equivalent to tapping in both tapes alternatively, as we described in the original scenario of the flanging effect, but giving an extra tap or slow to one of the tapes, creating a "permanent" gap between tapes under the actual fluctuation of r(t).

Deriving this function to the frequency domain, we will get the transfer function this filter represents. The entire procedure is indicated in [5].

Starting with the time domain formula and replacing r(t) with τ, the frequency domain equivalent expression is

$$Y(Z) = X(Z) + Z^{-\tau} X(Z)$$

Then, with regards to the transfer function H(Z)

$$H(Z) = \frac{Y(Z)}{X(Z)} = 1 + Z^{-\tau}$$

Applying the $Z = e^{j\omega}$ equivalence

$$H(e^{j\omega}) = 1 + e^{-j\omega\tau}$$

Switching the 1 to base e we will be able to find helpful combinations to express this equation in terms of sinusoidal functions

$$H(e^{j\omega}) = e^{-j\omega\frac{\tau}{2}} (e^{j\omega\frac{\tau}{2}} + e^{-j\omega\frac{\tau}{2}})$$

Finally applying Euler's formula $\cos(x) = Re(e^{ix}) = \frac{e^{ix} + e^{-ix}}{2}$ we find a more intuitive expression

$$H(e^{j\omega}) = 2e^{-j\omega\frac{\tau}{2}} \cos(\frac{\omega\tau}{2})$$

To clean the final expression getting rid of e-based components of the equation, absolute value will be applied (f_s= sampling rate (Hz))

$$\left|H(e^{j\omega})\right| = 2\left|\cos(\frac{\omega\tau}{2})\right| , \ \omega = 2\pi\frac{f}{f_s}$$

Changing the values of τ, the frequency response shows variations on the location of the notches. These notches are uniformly spaced ($1 / \tau$ Hz), and the first notch is placed at $1 / 2\tau$. Due to these variations of τ, the quantity and location of these notches are affected. In order to display the effects of the variations, a MATLAB code has been

implemented using two different values for τ of a significant difference. The code and the graphs obtained are shown below.

```
%____Initial_values_____
tau_1 = 4;
tau_2 = 20;
step = 0.01;
fs = 2;
index = 1;

%____Vectors_____
H = zeros(1,index);
w = zeros(1,index);

%____H_values_____
for f = 0:step:(fs/2)
    w(index) = 2*pi*f/fs;
    H(index) = 2*abs(cos(w(index)*tau_1/2));

    index = index + 1;
end

plot(w, H,'DisplayName', 'tau = 4')

xticklabels({'0','\pi/\tau','2\pi/\tau','3\pi/\tau','4\pi/\tau','5\pi/
\tau','6\pi/\tau'})

grid on
```

The idea of the code is to create two arrays, one for each variable ω and H. To create these two arrays, an array of 1x1 is created for each variable. Once the 'for' loop is running, each array will save the value of their assigned operation and will create a new slot in both arrays in every iteration.

Respecting Nyquist-Shannon sampling theorem, the value of 'fs' has to be at least double the value of 'f' to ensure all the information is captured along the sampling process. With a value of 0.01 for the "step" variable, a sampling frequency of 2 is enough to get a well-defined representation of H in the frequency domain. As can be seen in "H values" section of the code, the frequency axis is sampled step by step, collecting each sampling value of ω and H for a later representation of $\mathrm{H}(e^{j\omega})$.

For the τ = 4 case the distance between notches is significantly spaced. Only two notches can be seen, while first notch occurs at $\frac{\pi}{4} \simeq 0.785$(rad), being 1.57(rad) the distance between two consecutive notches.

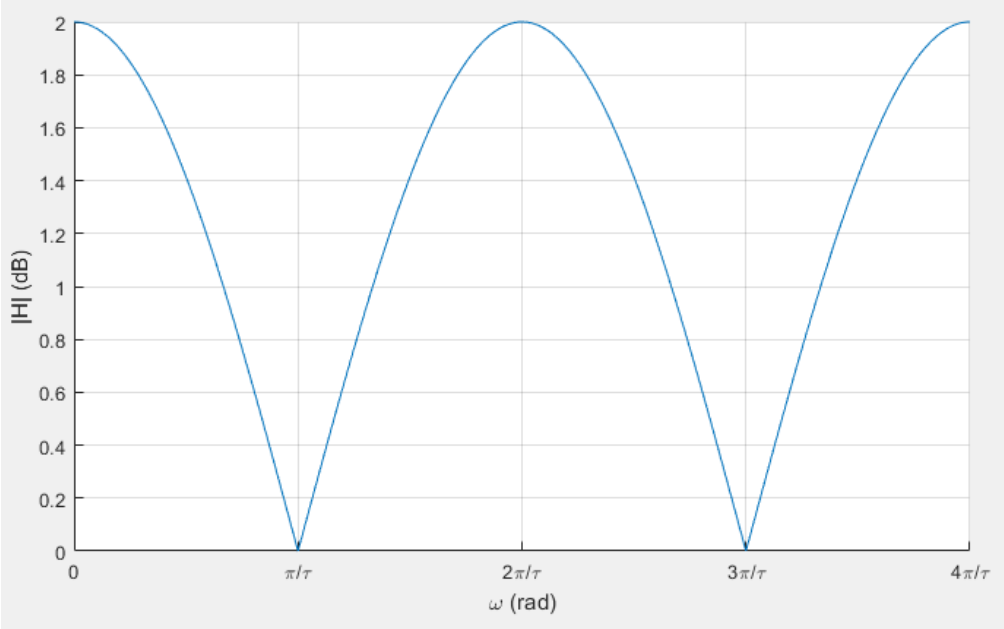

Figure 17. Magnitude frequency response of a flanger filter with τ = 4

On the other hand, a high value of τ ($\tau = 20$) causes the notches to be much more close to each other, as we can see in the figure 18. The location of these continues to be in odd multiples of $\frac{\pi}{\tau}$.

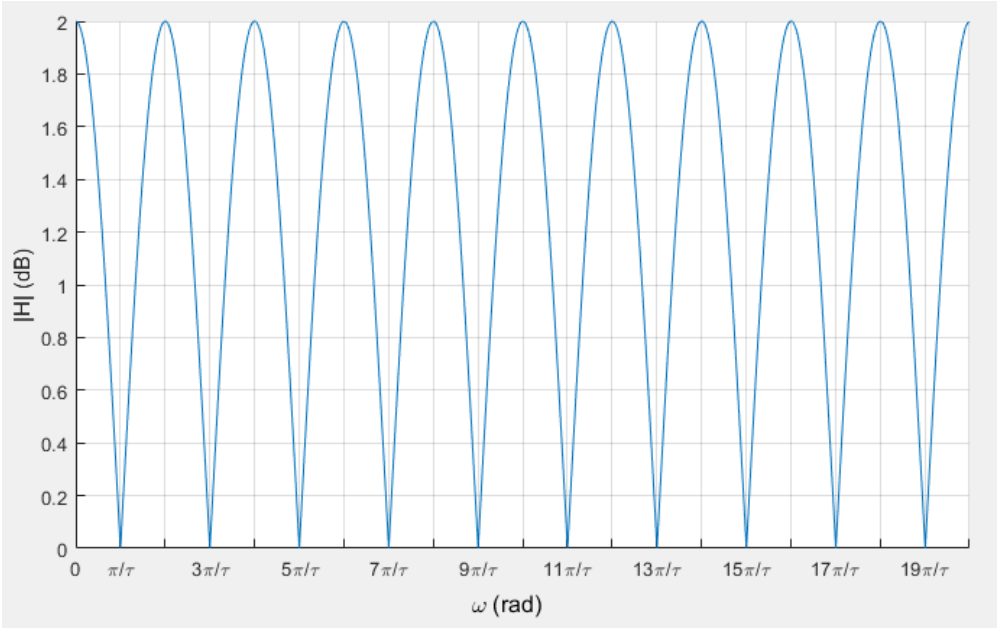

Figure 18. Magnitude frequency response of a flanger filter with τ = 20

In order to visualize the effects produced by a flanger filter on a real audio track, a flanger has been built in Simulink (figure 21) following the schematics shown in figure 22. After simulating it using pink noise as the input signal, the output signal of the Simulink implementation is clearly represented in the comparison between Figure 19 (dry) and Figure 20 (wet).

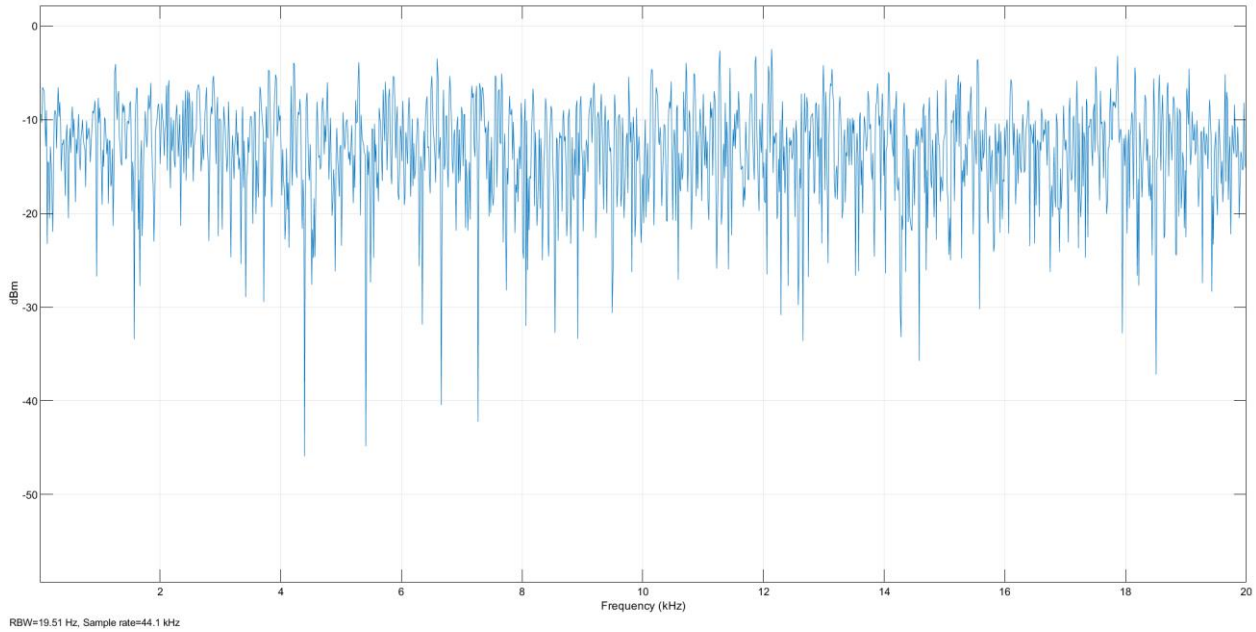

Figure 19. Original spectrum of the input signal (white noise)

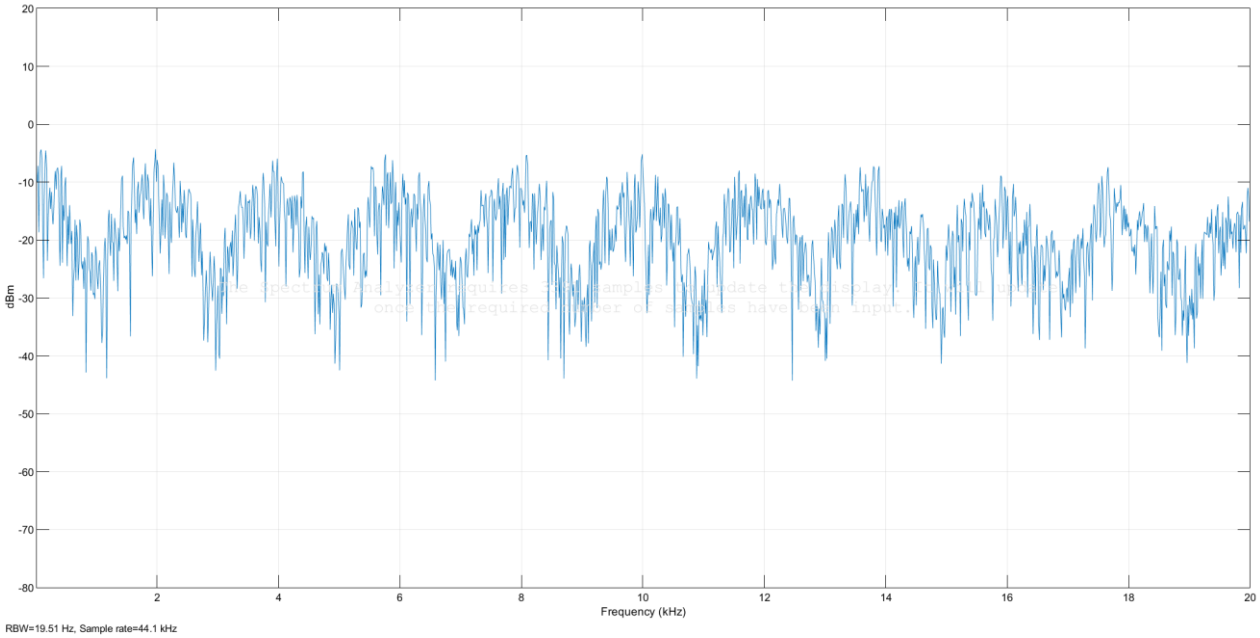

Figure 20. Spectrum of the white noise signal at the output of the flanger filter implementation using Simulink.

As predicted in previous MATLAB implementations' –see figures 17 and 18-, a set of uniformly spaced notches appears once the pink noise wave has been filtered through our Simulink implementation. Even though the notches keep a uniform separation between themselves, it is crucial to mark that, in a real time implementation, this uniform space between notches grows and shrinks with time -maintaining the uniformity-, creating the characteristic flanger sound. As it will be presented in the next section, different implementations of the flanger will lead to slightly different visualizations of these notches.

2.2.3 Structure

Unlike the phaser, there is no need of a filter to modify the signal in order to create the flanging effect. In this case, a delay block is needed, as well as a low frequency oscillator (LFO) to control this delay block and a couple of gain blocks.

The function of the delay block, as it name says, is to set the minimum time of delay the modified part of the signal is going to suffer. In the frequency domain, this block adjusts the distance between notches: the smaller the delay, the further apart notches will be and vice versa. This is demonstrated in the figures of the section 2.2.2, being τ the time delay applied at the delay block.

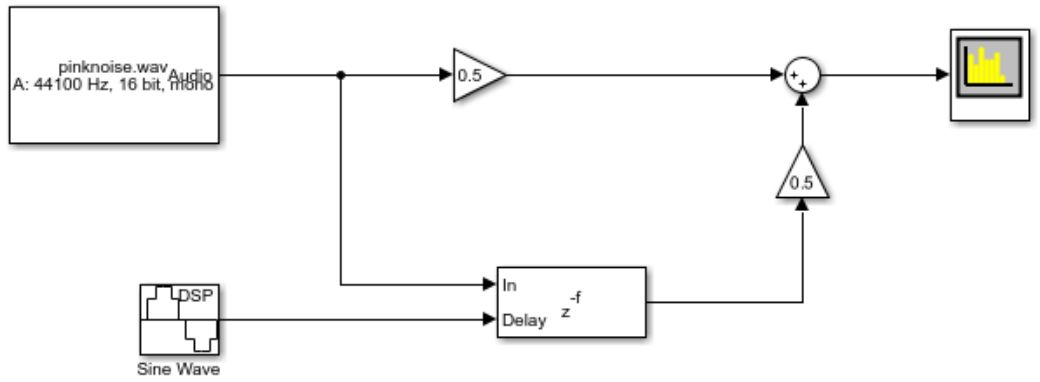

Figure 21. Flanger implementation created by the student in Simulink, using the implementations studied in previous section.

Another important block from the flanger structure is the LFO. As said before, the delay on the flanger is not a fixed delay, it is constantly variating by the control of a LFO. Also, this LFO have some parameter that are key on how the flanger filter is going to modify the input signal. To begin with, the amplitude peak to peak of this block will determine the range of delay applied by it; the greater peak to peak amplitude, the greater variations on the delay applied by this block. A big peak to peak amplitude will cause a remarkably audible flanger effect.

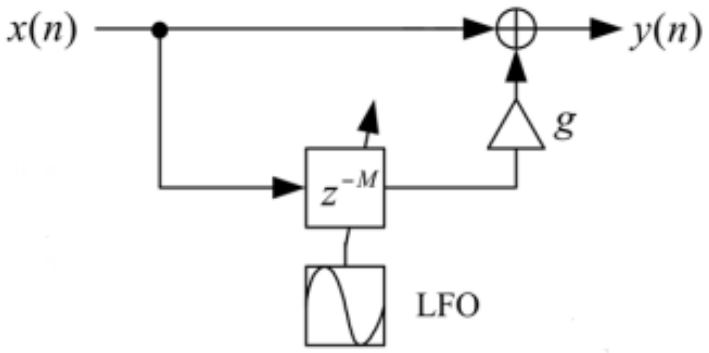

Figure 22. Basic design of a flanger filter. [12]

The gain block g is used to control the amount of delayed signal that is mixed with the non-delayed part. With a value of zero, the signal on the frequency domain would be constant. However, as it grows up to one, notches start to appear. These notches will reach the value of zero if the value of g = 1. As the flanger audible effect is produced by those notches, the closer to unity gain at *g*, the more noticeable will be the flange effect.

The reason for this value of g is no other than a balanced mix of both parts of the signal. As the non-modified part has no losses, it already has the unity gain. As both parts of the signal are equally necessary, setting different gains on each path will lead to a poor sound of the flanger effect, even if the delayed part of the signal has a gain greater than unity gain.

In the code implementations of previous sections, it was shown that modifying the value of the delay block it was possible to obtain a different quantity of notches in the frequency domain. In the block design presented the appropriate parameter to control that quantity is the amplitude of the LFO. For a higher amplitude, more notches will appear.

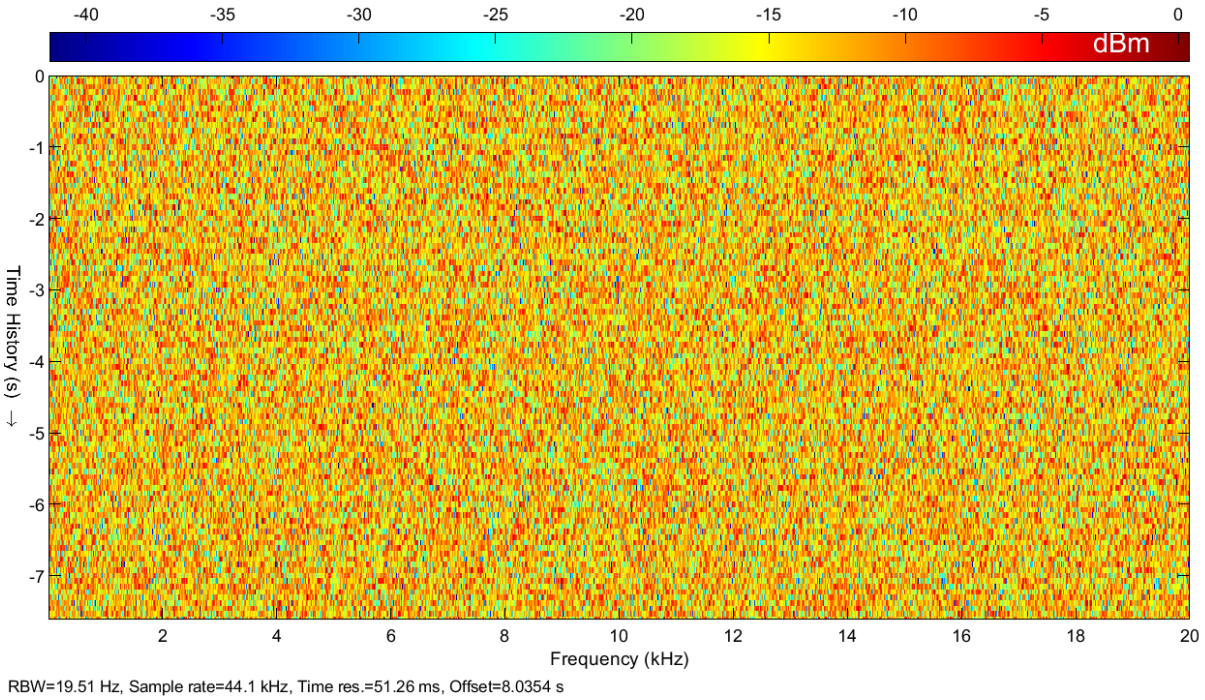

Figure 23. Spectrogram of the white noise input of the flanger.

Many nowadays implementations of the flanger include a feedback loop around the delay block, as implemented in Figure 24. This feedback loop includes a gain block *f* to control the amount of fed back signal.

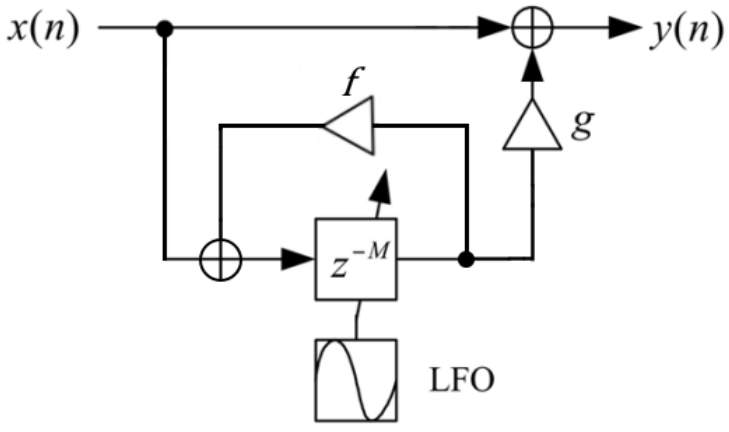

Figure 24. Flanger structure with feedback loop. Figure by Vesa Välimäki.

This loop adds a clearly audible effect due to the creation of a feedback comb filter with positive added in the future to the existing feedforward comb filter. Based on that added loop, the output of the flanger filter is appreciably changed, as the notches created acquire a sharp waveform. With regard to the spectrogram, the shown shapes will be similar, only changing the depths of the waves due to the sharpness of the notches acquired from the feedback loop.

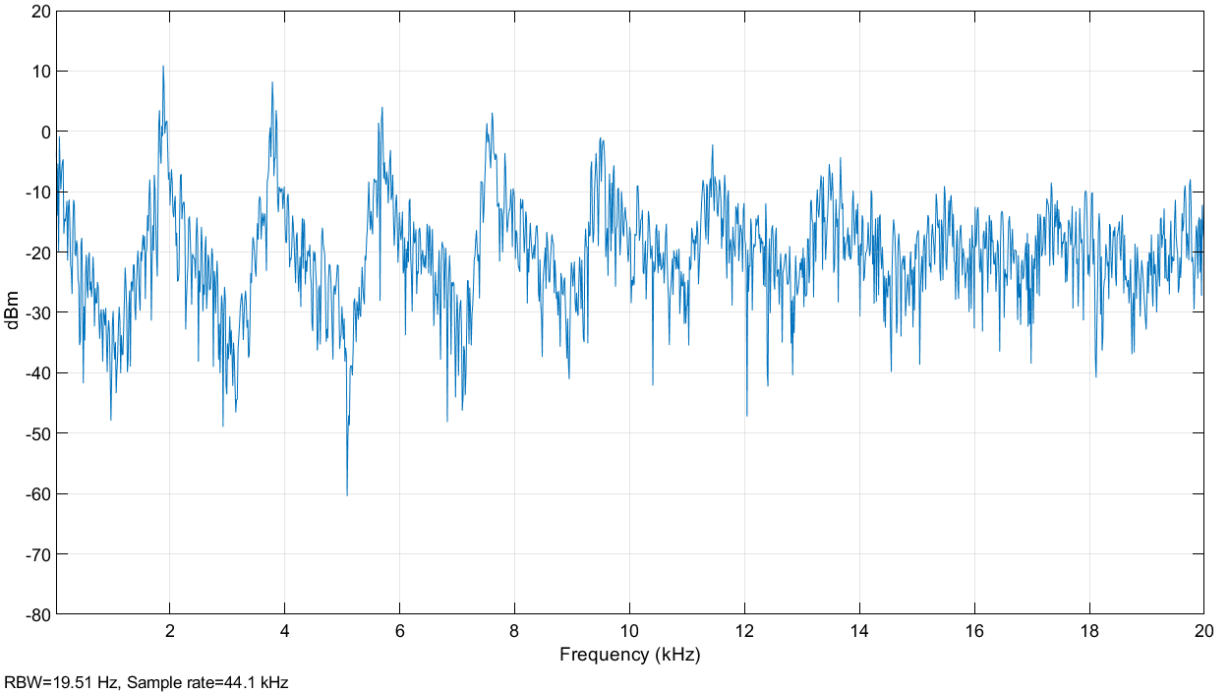

Figure 25. Spectrum of the white noise signal at the output of the flanger filter with the additional feedback loop. f = 0.95

As the feedback loop could not be added to the student Simulink implementation due to an algebraic loop failure, another pre-implemented Simulink flanger was used, including this block the feedback loop.

Using the spectrum analyser tool configured to show the spectrogram, clearly wave patterns appear after the effect of the flanger on the white noise input (Figure 26). Figure 23 shows the spectrogram output of the non-modulated white noise input.

The spectrogram with the feedback loop added to the flanger structure seen in the Figure 24 does not change the situation or shape of the notches, but with the results seen in Figure 26 we can tell that the depth of these notches will be affected, making the colours representing the dBm values stronger in both ways.

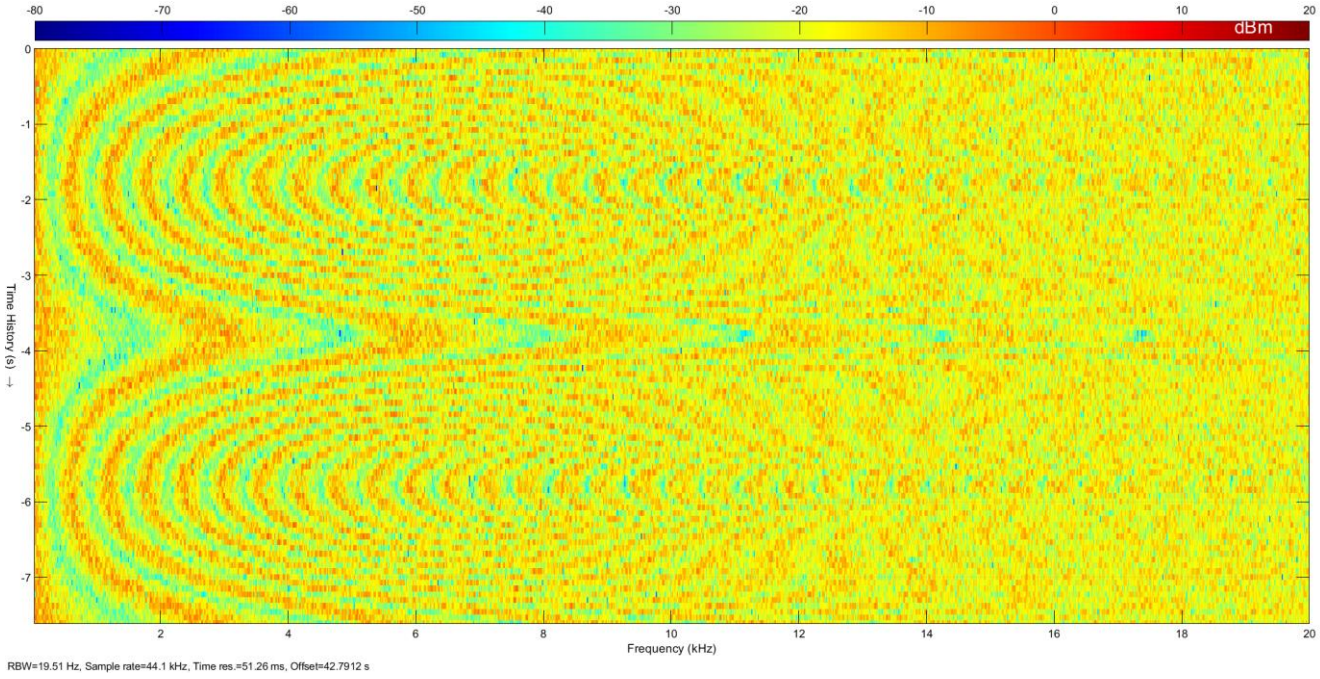

Figure 26. Visible patterns on the white noise spectrogram at the output of the flanger.

2.2.4 Variations

In this section of the project, some flanger-related algorithms will be reviewed, as they can provide a wider vision when it comes to comprehend the mainly studied filters. Thereby, the overall attempt for this section is to study the next effects not as deep as the previous ones, but to achieve some level understanding of the performance of them, as some are simply modifications of our main effects and others are parallel effects with interesting hardware implementations. All the effects below will be presented to the lector using my own words, aiming for a natural but concise definition of each algorithm.

2.2.4.1 Chorus

Thanks to this fulfilling effect, one single instrument can sound like multiple instruments playing the same notes at the same time, just like an instrument chorus. To reach that sound, the input signal will be multiplied for a later mixing of all the resulting signals. However, if all the signals are added without any modifications, the output signal would be the same as the input one, louder. So, to understand what happens with the chorus effect, a description of an example of what this effect will sound like will be presented.

A real scenario of this effect will be presented with the most typical example: imagine two or more musicians playing the same melody simultaneously. Why, in this case, the resulting sound is not the same riff but louder? Although all the musicians will try to play same notes at the same time, human imperfection will not let that happen. Many factors will make those guitar sounds sound in a different way; even though the main factor would be the random retards all musician will introduce in the sound. If this example is used with electric guitars as the instruments played, many factors can be mentioned as to prove the differences between all of them. The retards introduced will be produced by all guitarists, as the same guitarist will be anticipated and delayed randomly along the riff. Also, each guitar will have a slightly different tuning, as well as different strings. All this random factor will lead to a not-so-clean adding of all the riffs played, creating a complete and fulfilling sound effect. This is the effect created by the chorus algorithm replicates.

The most common implementation for this effect would require the use of low-pass noise to control the delay of each line. Filtering a noisy signal can bring a clear random wave that could easily control the amount of delay of the delay block. This way, the delay would be generated randomly and therefore in a more realistic way, approaching the guitarists example.

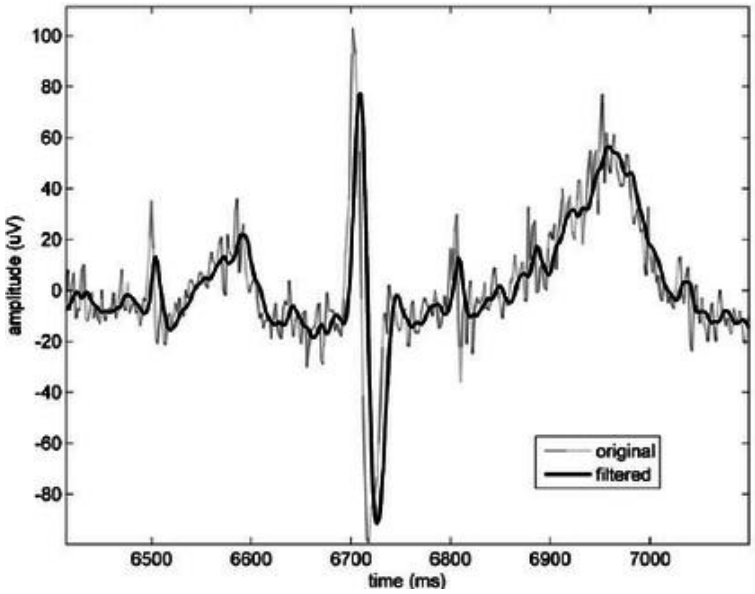

Figure 27. Example of filtered noise.

As a flanger variation, it can also be composed by a LFO controlling a delay block, but the main difference with the flanger resides in the quantity of delay lines that this filter has. As we saw before, the schematic of a flanger consist of an unmodified (dry) signal mixed together with a delayed (wet) version of itself, changing this delay length with a LFO. In the chorus, instead of just one delay line, there is more than one delay line and each of them is controlled by an independent LFO. Also, the delay introduced by the chorus is usually greater than flanger's delay. In this LFO case, the frequencies used should be under 3Hz in order to avoid fast compressions and decompressions of the modified signal, leading to significant detunes (unless that is what the user is looking for).

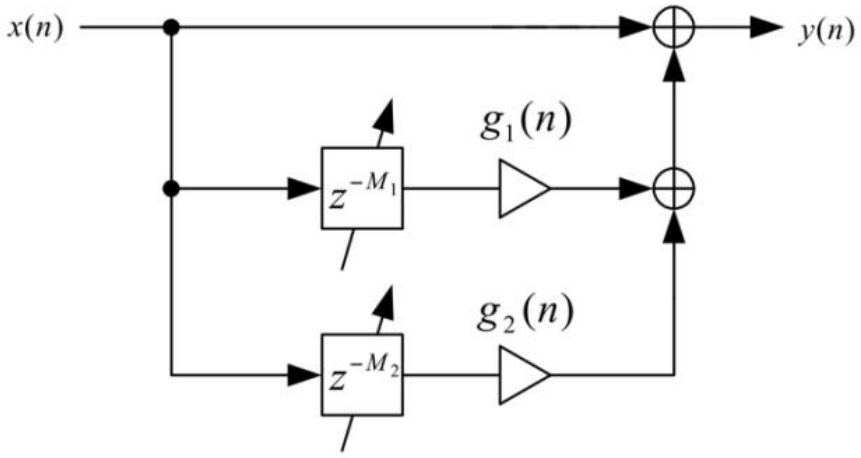

Figure 28. Chorus block diagram. Figure by Vesa Välimäki.

The time-domain formula modulating this schematic would be:

$$y(n) = x(n) + g_1(n)x(n - M_1) + g_2(n)x(n - M_2)$$

Being g1 and g2 gain blocks controlling the quantity of each voice that will be mixed with the original signal, changing this way the chorus effect power.

2.2.4.2 Through-zero flanger

Although the implementations of the flanger seen in the previous sections are the most common ones, they also are very basic, bringing some problems when used. With naïve implementations, the dry (non-modified part of the signal) and delayed signal will never coincide. Actually, when the flanger is created with analogic tapes, as originally created, both of the tapes suffered a variation on their speed in order to catch the other

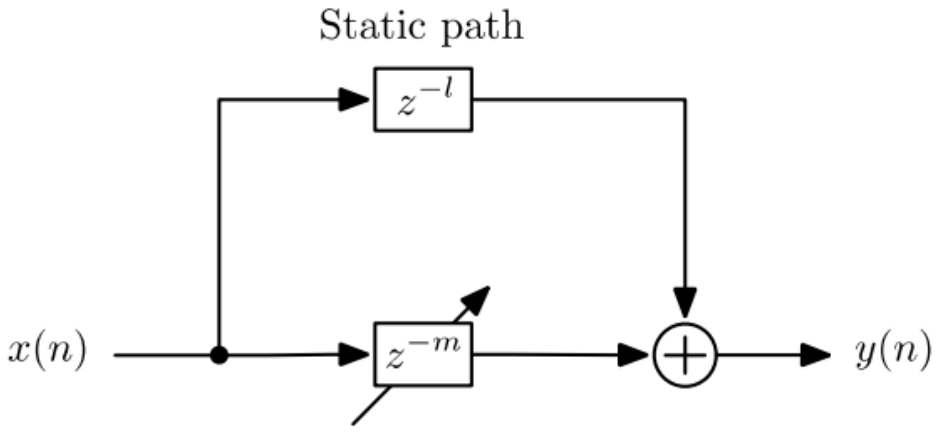

Figure 29. Structure of the through-zero flanger. [12]

tape delay. In other words, at some point both signals were again on perfect alignment, and then one of them continued to slow down, creating a more noticeable flange effect.

With the basic flanger implementation, the unmodified part of the original signal goes directly to the adder at the last stage of the filter. For this reason, there is no physical way to add a previous part of that original signal to that unmodified part, as it will always be faster than the delayed part, even in the minimum-delay-scenario.

In other words, if both paths are seen as both flanges at the original scenario, modelling the flanger as in previous sections (2.2), one of the flanges would never be delayed. So, once a flange has been tapped and therefore delayed, the flanges would never synchronize again. The solution for this problem relies on the through-zero implementation. Here, both flanges are delayed, one of them with a variable delay and the remaining one with a static delay. If the variable delay takes a small enough value of delay -smaller than the static delay-, both flanges will synchronize again.

This can be achieved in the virtual model by adding a static delay "L" to the dry signal, being $L \simeq M_{max} / 2$. This resulting adaptation of the classic flanger structure is called through-zero flanger.

2.2.4.3 Barber-pole flanger

With special configurations of both flanger and phaser filters, an effect of infinite rising or falling notches can be seen in the output signal spectrum for both kind of filters. In this section, some ways to reach this effect will be discussed regarding F. Esqueda, V. Välimäki and J. Parker paper [11]. The fist method, introduced by Shepard in the 1960s, consists on a series of notch filters separated in the frequency domain one octave apart from each other.

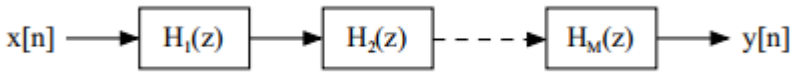

Figure 30. Barber-pole filter, Shepard structure [11]

The second method uses two flangers in cascade with independent LFO for each delay block, and the third method implies single side band modulation, known for its frequency shifting.

Using the first method and representing the resulting signal in a logarithmic scale, when the center frequency of the notches are settled at one-octave intervals, a set of uniformly spaced notches can be seen. In the figure presented, the attenuation of the notches is controlled by an inverted function proposed by Shepard.

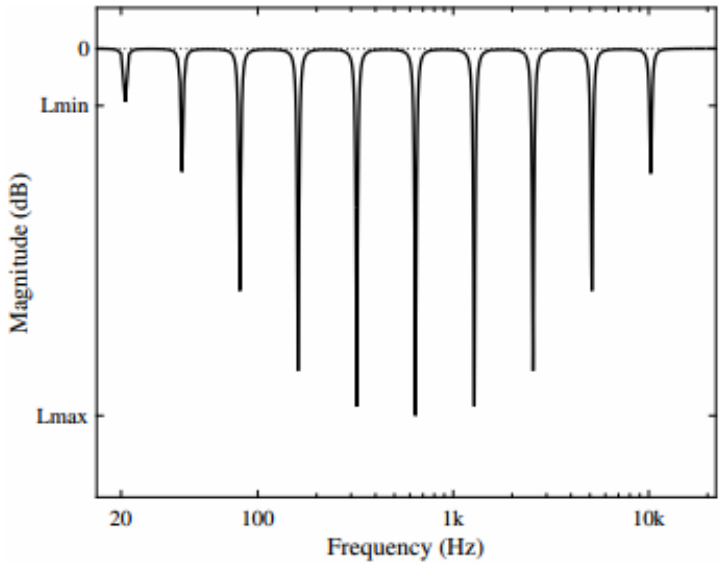

Figure 31. Magnitude response of 10 cascaded gain-modulated notch filters. [11]

For this case, 10 allpass filters were used, and f_0 = 20Hz. The repetition rate of the effect is ρ = 0.1Hz. After a 30 second simulation with those parameters, the resulting spectrogram is presented in the Figure 32. The three white triangles on the top of the spectrogram represent the start of each cycle.

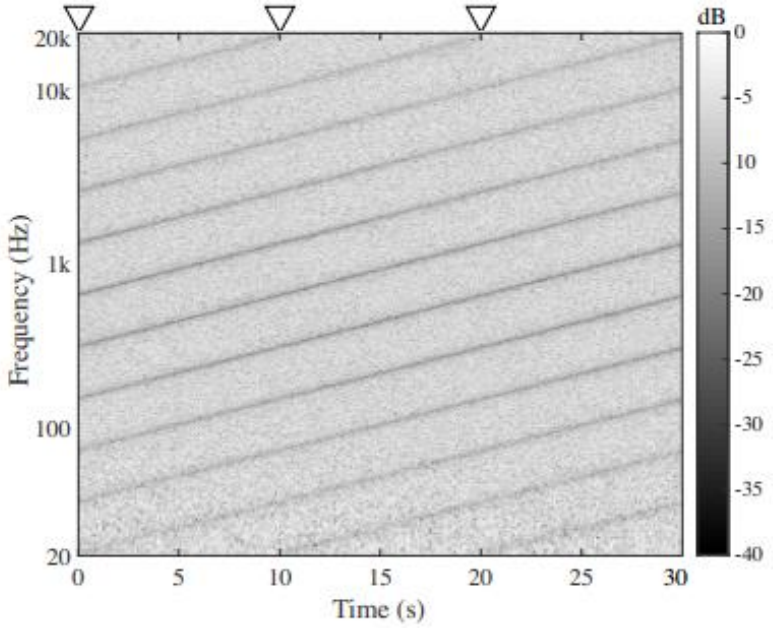

Figure 32. Spectrogram of a barberpole filter implementation with 10 cascaded varying notch filters. [9]

Regarding Figure 32, it is possible to appreciate how new notches appear from the bottom of the spectrogram as the ones approaching the Nyquist frequency disappear. Also, as presented in Figure 31, notches situated in the middle frequencies are deeper than those situated at the top and bottom frequencies.

As seen in the previous implementation, the number of notches used in Shepard implementation is constant. When using the second method, flangers cascade, the number of notches varies over time due to the flanger's functioning. As the objective of this implementation is to simulate a barber pole illusion in the spectrum of the filtered signal, it is necessary to ensure that the notches movement is unidirectional. The solution presented in [11] deals with this problem. Controlling each delay block with independent

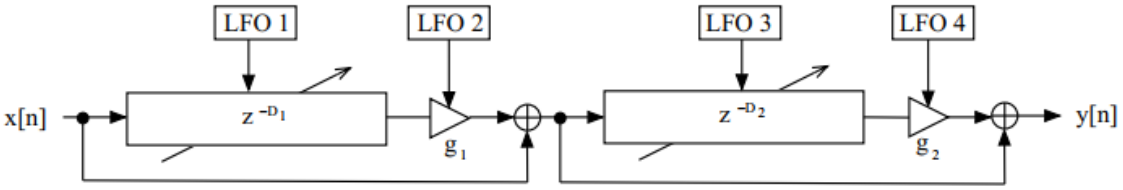

Figure 33. Dual flanger proposed system for a barberpole effect block diagram [9]

sawtooth waveforms LFO would solve the delay line reset. To implement the sawtooth function in discrete time, a modulo counter would reduce the aliasing that affects a simple sawtooth function in discrete time.

$$s(n) = (D_{max} - D_{min})[(n\Delta)\ mod\ 1] + D_{min}\ ,$$

where n is the time step index, D represents the delay length in samples, and $\Delta = \frac{\rho}{F_s}$ is the phase increment. With the formula presented, the effect would be perceived as a descending filter sweep. To change directions, s(n) need to be flipped horizontally:

$$s'(n) = (D_{max} + D_{min}) - s(n).$$

As the waveform has a sudden change in its amplitude, when this change affects the delay blocks, it will cause a slap on the output signal, interrupting the infinite loop effect. To avoid this, a second flanger structure will be added at the output of the first one and cross-fade will be used to switch between them. To achieve the cross-fade switch, each gain block will have an independent LFO controlling the gain value. The structure

will be identical, but second line LFOs have to be 90º phase shifted. Also, a triangle waveform would be a good choice for the implementation of the cross-fade.

A possible improvement of this structure implies adding fractional delay filters. This way, the movement of the notches in the frequency domain will be smooth. If the delay blocks use rounded values, the notches will move in steps, leading to a "zipper noise" output.

The resultant spectrogram on a 30 second simulation with $D_{max} = 66$ and $D_{min} = 44$ and $\rho = 0.1$Hz. The spectrogram of the Figure 34 is from [9]. In that case, a third-order Lagrangian fractional delay filter was used to accommodate fractional delay lengths. Contrary to the previous method spectrogram, the notches are not parallel lines all across the Figure. As they reach higher frequencies, they start to increase the slope until they appear as crossed lines, due to the low pass response of the fractional delay interpolator. The blurry horizontal lines represent the point where the two flangers meet while cross-fading.

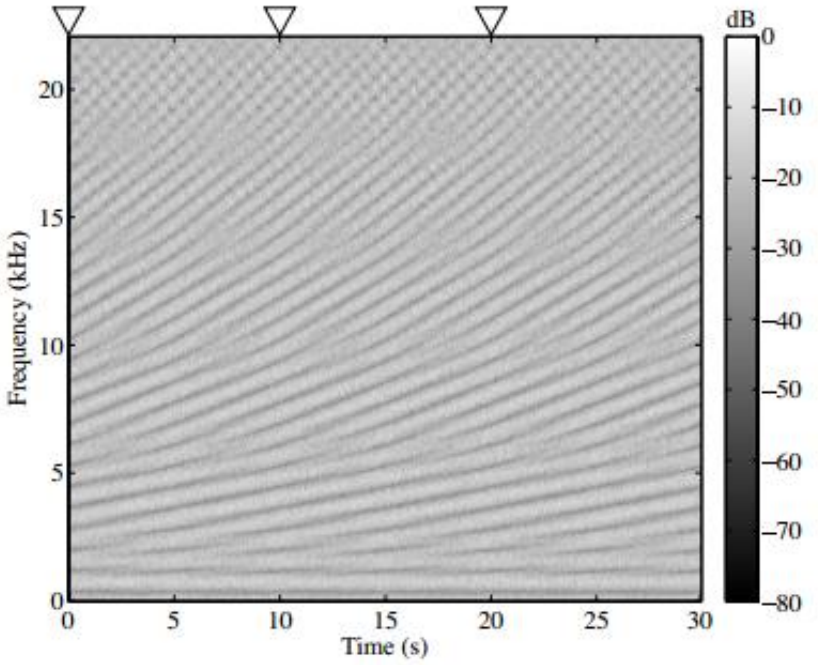

Figure 34. Spectrogram of a barberpole effect implemented with a flanger cascade and white noise as the input signal. [11]

The last method consists on single-sideband modulation, and it was first proposed by Harald Bode. The idea is, once again, using the mixing of the modulated signal and the same signal at its original form, producing notches whose movement is controlled by the frequency-shift applied. The expression for this method would be:

$$\sin(\omega t) + \sin(\omega t + \omega_0 t) = A(t)\sin[\omega t + \phi(t)],$$

being ω the frequency of the sinusoid and ω_0 the amount of frequency shift. Obtaining the value of $A(t)$ from the previous expression:

$$A(t) = \sqrt{2 + 2\cos(\omega_0 t)},$$

obtaining an amplitude independent from the frequency ω.

To create a frequency-dependent amplitude, the only thing needed is a delay time added in the modulated signal. That way, the expression would be:

$$\sin(\omega t) + \sin[(\omega + \omega_0)(t - \tau)] = A(t)\sin[\omega t + \phi(t)],$$

solving A for:

$$A(t) = \sqrt{2 + 2\cos(\omega_0 t - \omega_0 \tau - \omega t)}.$$

This way, the phase of the amplitude is depending on the frequency. Now it is possible to control the number of notches with the delay time τ and the movement direction with ω_0 in a SSB barberpole implementation like the one in Figure 35.

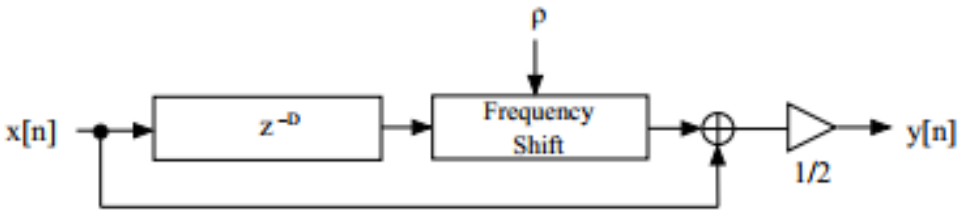

Figure 35. Generalized SSB-modulation based barberpole block diagram. [9]

With ρ, notches rate and direction are controlled, and the delay length D control the number of notches M by M = $\frac{D}{2}$.

With the barberpole implementation represented in the previous figure and applying a white noise signal in the input, the response of the system is shown in Figure 36. Even though the spectrogram shows a well-organized set of notches, the movement effect of those is not as convincing as the one seen in the first method implementation.

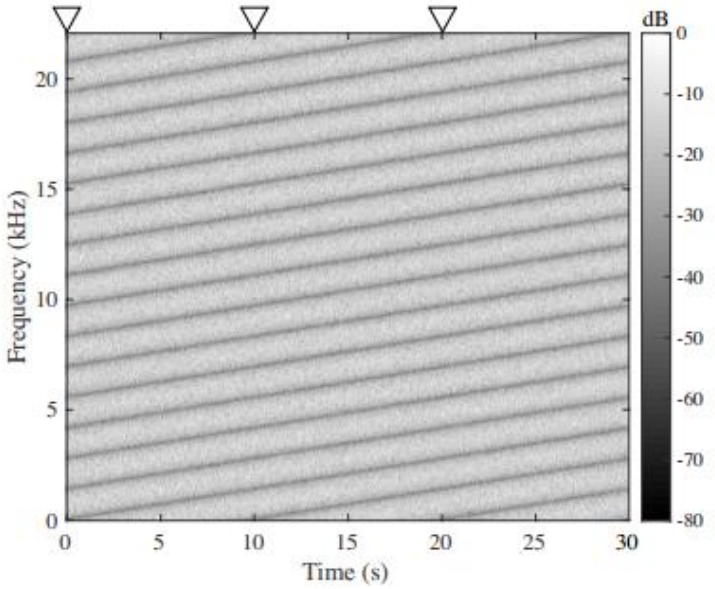

Figure 36. Generalized SSB-modulation based barberpole effect sprectrogram. [9]

As seen in the formula $A(t) = \sqrt{2 + 2\cos(\omega_0 t - \omega_0 \tau - \omega t)}$, if we want to vary the space between notches we need to make τ dependent of ω, and the way to do this is adding a delay in the frequency domain. As discussed in previous sections, this can be achieved with a first-order allpass chain, with a transfer function given by

$$A^D(z) = \left(\frac{a + z^{-1}}{1 + az^{-1}}\right)^D$$

being D the number of allpass filters composing the chain. Even though the delay block has been replaced with an allpass chain, the number of notches M is still given by $M = \frac{D}{2}$.

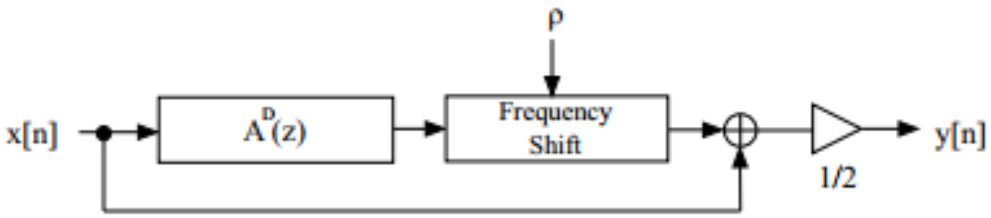

Figure 37. SSB-modulation based barberpole block diagram using spectral delay filter block diagram. [9]

.

From the first method to create the barberpole effect, we know that the ideal distance between notches is given by octaves. To achieve this separation, the component a from A^D has to be negative, yet not meant to take big negative values. This could also be achieved by optimizing the first order coefficients of each allpass filter composing the chain.

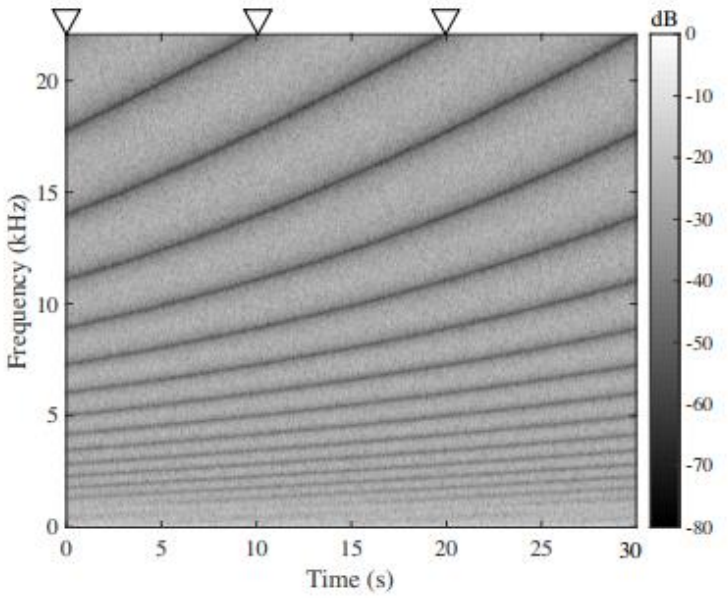

Figure 38. . SSB-modulation based barberpole spertrogram [11]

Now, after the change applied for a better performance of a barberpole effect, the output for the new implementation is given by the Figure 38.

2.3 Leslie Speaker

The Doppler effect as well as the Leslie speaker will be presented in this section, together with an efficient algorithm to simulate them. To talk about the Leslie speaker, it is necessary to refresh the Doppler effect, as this speaker is mainly based on that principle.

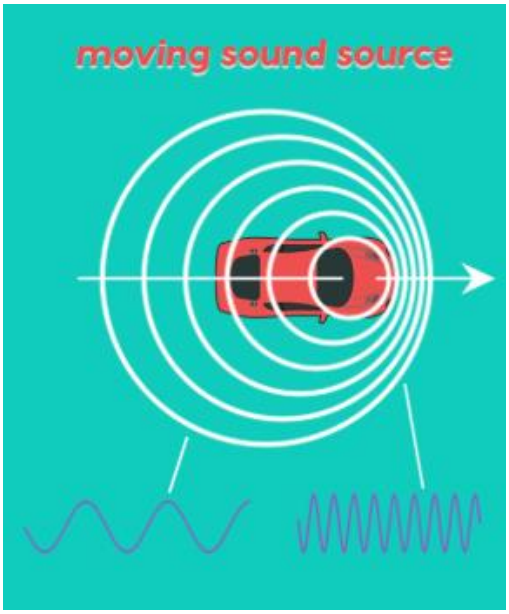

Figure 39. Doppler effect schematic example.

Discovered by Christian Doppler in 1842, Doppler effect causes a frequency change on a sound wave due to the movement of the audio source and/or the position of the listener. A common example of this effect can be heard when an ambulance is passes by the hearer at some speed. The movement of the ambulance, being the audio source, creates a compression of the sound waves produced in the same direction as the ambulance's and, at the same time, sound waves produced in the opposite direction are de-compressed.

The Leslie speaker characteristic sound is created by the rotation of both the horn and the woofer unit seen in Figure 40. It receives the name after its inventor, Donald Leslie. As seen in the Figure 40, the rotating horns and woofer are implemented inside of a box, causing the Doppler effect generated to be also modified, creating different shifts of it and thus, producing a chorus like effect.

The simulation of the Doppler effect can be achieved using two delay lines controlled by signals delayed $\frac{\pi}{2}$ radians. These delay lines can be seen in Figure 41. In order to simulate the directional horn's sound, the output amplitude of the delayed lines will be controlled with sinusoidal waves. When controlling the amplitudes mentioned, we must keep in mind that this modulation has to be synchronized with the signal controlling the delay blocks, so the horn moving to the back has a lower tone and its amplitude is decreasing. As we can see in the Figure 41, the unequal modulation mixing of the signals creates the stereo effect.

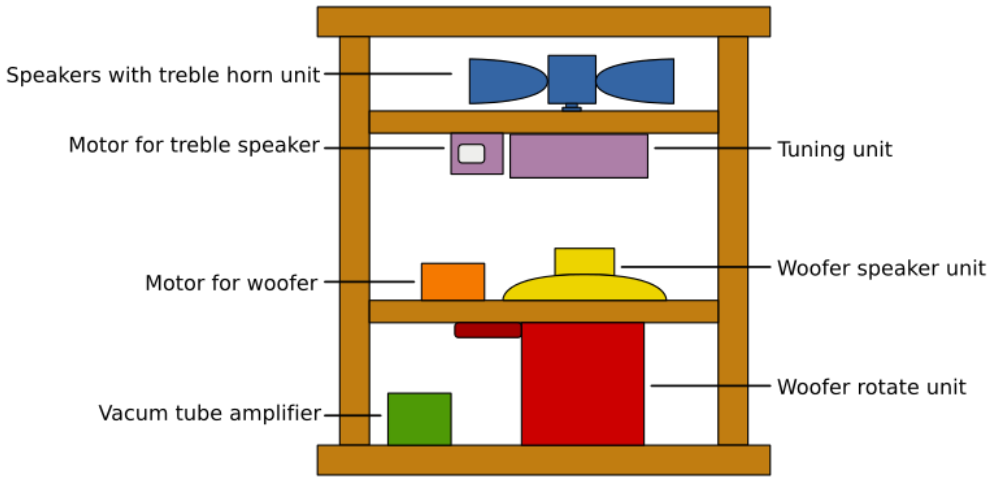

Figure 40. Leslie speaker diagram [Wikipedia]

To simulate the independent rotation of the horns and woofer a low pass and a high pass filter will be used to separate the high and low frequencies of the signal. As the frequency rotations are different for each part of the signal, two VCOs will be used, one for each block.

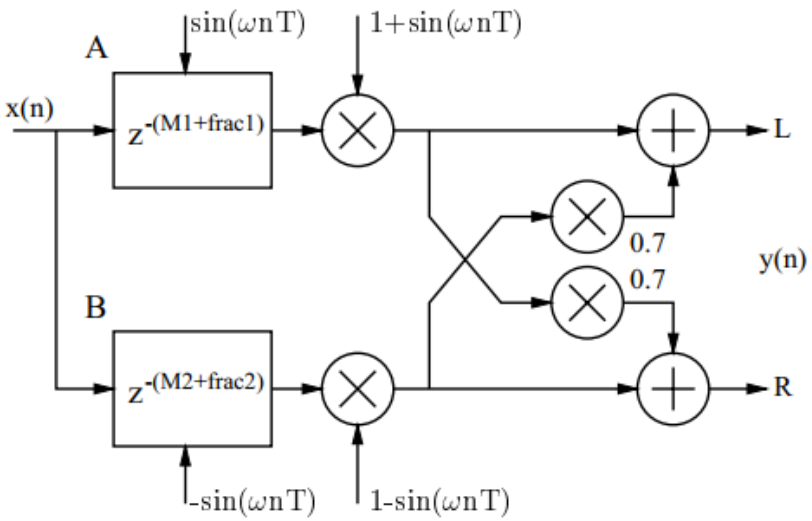

Figure 41. Simulation of rotating speakers. [15]

As Leslie speakers usually have two rotating frequencies – one for the choral effect and another one for the tremolo effect, the switching process between these two modes is simulated by the ramp signal presented in each VCO, shown in Figure 42.

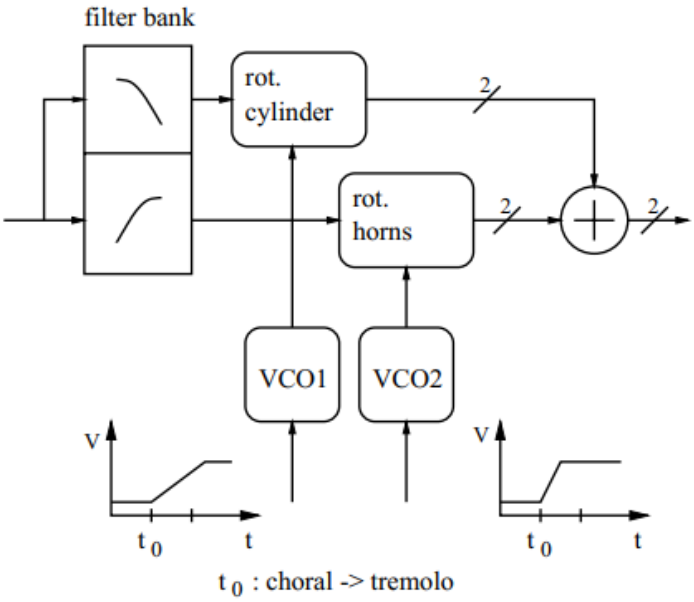

Figure 42. Rotary speaker implementation. [15]

3. CONCLUSIONS

To complete this bachelor thesis, a research work through all the issues and working lines has been done, achieving a deep understanding of all the topics presented in this project. Matlab and Simulink implementations have been very helpful in that way, as they have given me the basics on how to implement each algorithm and the effect they produce in the input signals. Hardware implementation has also been very challenging and useful, as it has helped me to learn the specific implementation of each component of the block diagrams for a musical field.

On the basis of my previous trial and error use of the effects studied in this project, the knowledge acquired studying and working with each algorithm has given me a big picture of the characteristics and performance of each effect. This big picture acquired has been used to create a big overview of the ideal application and operating environment to maximize the performance of these music technology algorithms.

Also, the appropriate process for digitalizing an analogic system through the composition of the system itself and the role of each component has been very challenging, as I had to achieve a certain level of understanding for each process to be able to deconstruct the effect in code lines and functions with a similar outcome.

Each effect has been analysed from the history of its analogic system to the correct functioning of its implementing system, facing the subsequent simulation through software. For example, I have learned to avoid some problems related with the use of the flanger, working with distortion, as clearer signals can have coinciding harmonics with the notches created by the flanger and, this way, failing to create an output response.

Although the software and electronics used for this project have been rudimentary, it is clear to me that, with specific purpose applications, these effects can be implemented in software with very high quality outcomes, making them much more cheap than the conventional analogic implementations –yet not as precise as those last ones-.

4. FUTURE LINES

Based on the knowledge acquired along the process of creating this project, it could be continued using some useful and creative ideas. For example, once the audible effects have been linked to the implementations, it would be interesting to add modifications or even creating new custom effects based on a desired sound, always aiming for a complete understanding of the effects studied.

Hardware implementations of other effects may be interesting, regarding not only pure frequency based effects as the phaser, but other effects like flanging, chorus, through-zero flanger…

Also, it would be nice to design these effects in hardware description languages, such as VHDL, for a real time implementation in a FPGA.

Once both types of implementations would have been achieved, taking a quick look into the music/guitar pedal market would also be useful to get a big picture of possible competitors in the field. A great implementation on a device such as an FPGA would not be ignored, regarding a product where effects are sold in a virtual way. With this option, the user would only have to acquire the hardware device to store virtual effects, being able to select between many effects in an easy-to-handle way, reducing the costs of the physical production yet maintaining the versatility of a portable pedal set.

5. REFERENCES

[1] Hartmann, William M. "Flanging and phasers." *Journal of the Audio Engineering Society* vol. 26 nº.6 pp. 439-443 (1978)

[2] Bartlett, Bruce "A Scientific Explanation of Phasing (Flanging)." *Journal of the Audio Engineering Society* vol. 18 nº.6 pp. 674-675 (1970)

[3] J. Dattorro, "Effect design—part 2: Delay line modulation and chorus," *J. Audio Engineering Soc.*, vol. 45, no. 10, pp. 764–788 (1997)

[4] P. Liévano-Torres, J. Espinosa-Durán, J. Velasco-Medina "Algorithm Implementation in High-Fidelity Digital Audio Effects Using Programmable Hardware" *Diseño de un procesador multi-efecto para guitarra eléctrica basado en FPGA (code 2627) (2012)*

[5] J. O. Smith. "An allpass approach to digital phasing and flanging." In Proc. Int. Computer Music Conf. (ICMC'84), Paris, pp. 103–8, (1984)

[6] F. A. Bilsen, R. J. Ritsma "Repetition pitch and its implication for Hearing Theory" *Acustica.*, vol. 22, (1969)

[7] Wikipedia, "Flanging."

[8] Wikipedia, "Phasing."

[9] R. Kiiski, F. Esqueda, V. Välimäki "Time-variant gray-box modelling of a phaser pedal" *(DAFx-16), (2016)*

[10] J. O. Smith III, Lee, Nelson "Time Varying Delay Effects" *Center for computer research in music and acoustics (CCRMA),* (2008)

[11] F. Esqueda, V. Välimäki, J. Parker "Barberpole phasing and flanging illusions" *18th Int. Conference on Digital Audio Effects (DAFx-15)* (2015)

[12] V. Välimäki, F. Esqueda, B. Alary "Audio effects processing" (2019)

[13] F. Eichas, M. Fink, U. Zölzer "Feature design for the classification of audio effect units by input/ output measurements" *(DAFx-15)* (2015)

[14] U. Zölzer, "DAFX: Digital Audio Effects", 2nd Edition, (2011).

[15] S. Disch, U. Zölzer, "Modulation and delay line based digital audio effects", *2nd COST G-6 Workshop on Digital Audio Effects (DAFx99)* (1999)

[16] V. Välimäki, "Discrete-time synthesis of the sawtooth waveform with reduced aliasing," *IEEE Signal Process. Lett.*, vol.12, no. 3, pp. 214–217, (Mar. 2005)

[17] ElectroSmash.com "MXR Phaser 90 Analysis"

6. APPENDIX

Figures from the simulation of the filters studied on Simulink.

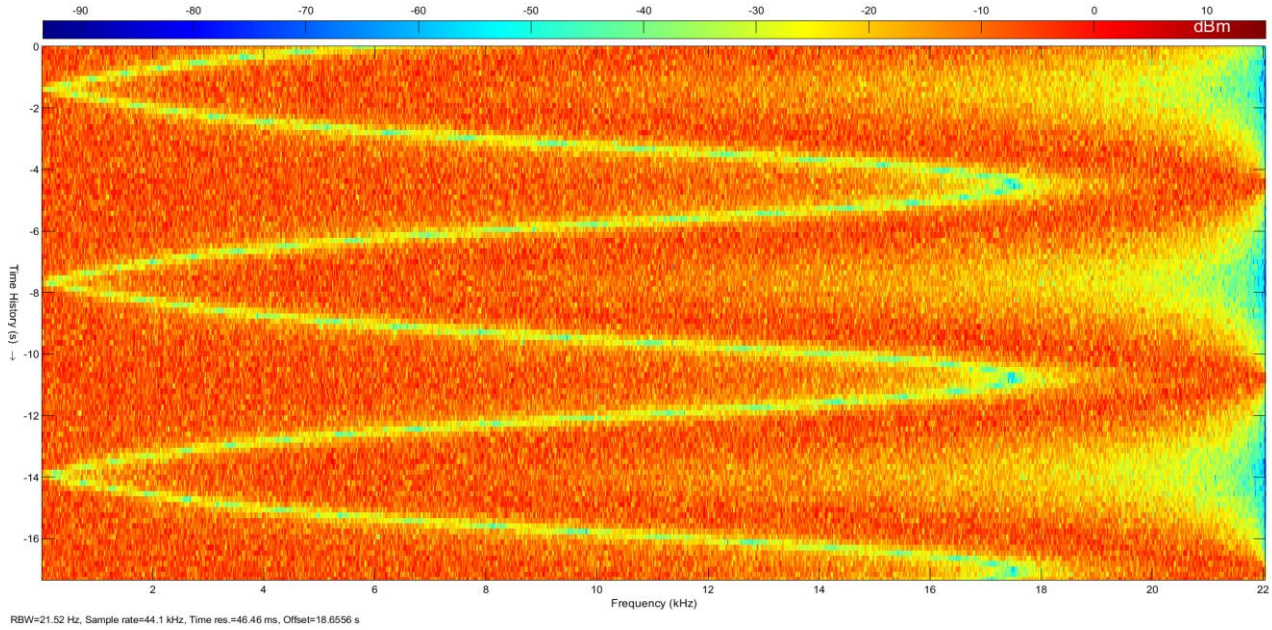

Figure 44. Phaser spectrogram of an implementation of one notch magnitude (two allpass filters) response and white noise as input of the system.

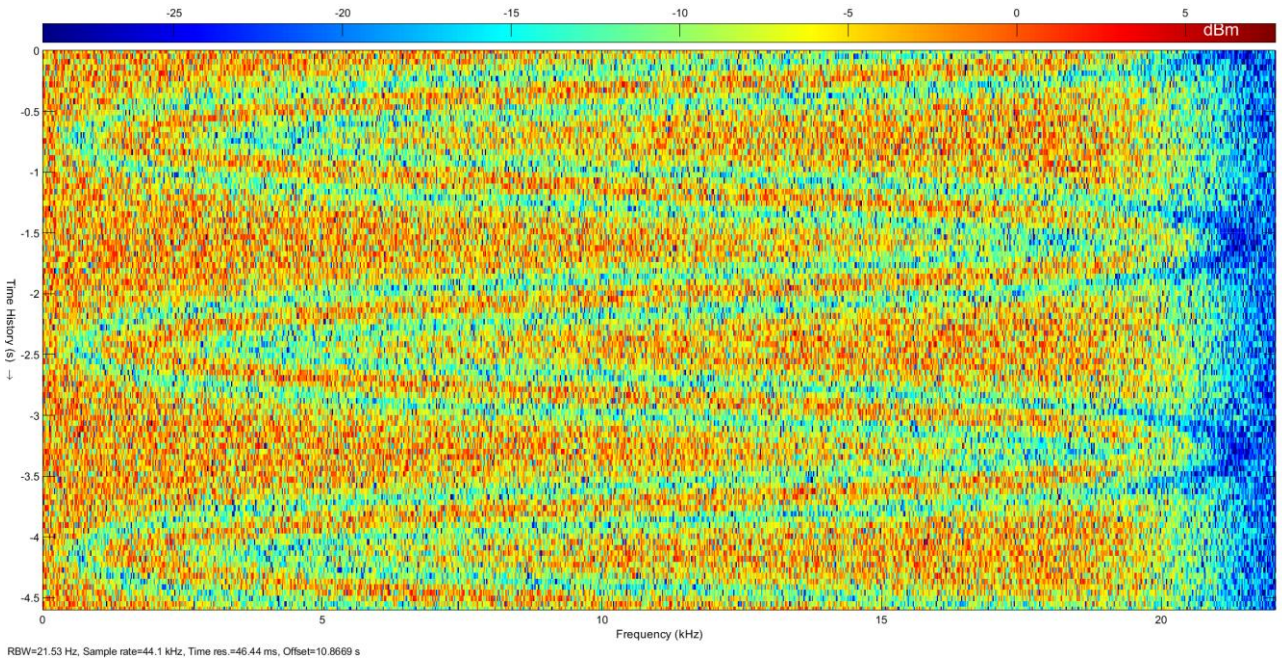

Figure 43. Phaser spectrogram of an implementation of two notch magnitude (four allpass filters) response and white noise as input of the system.

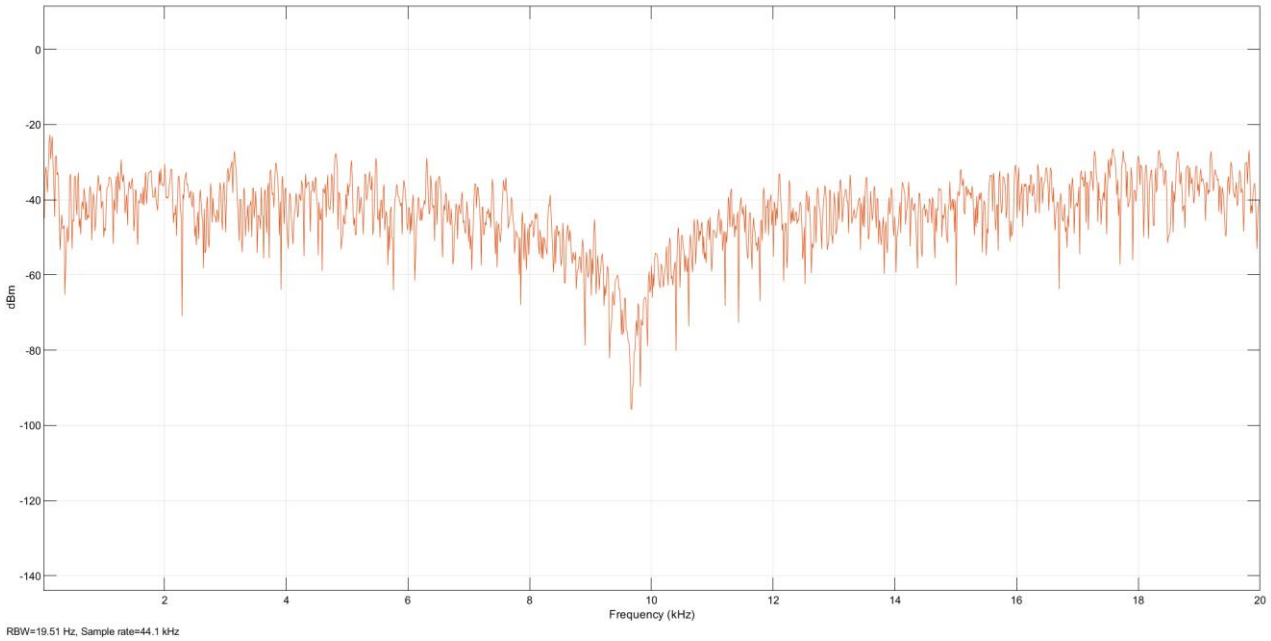

Figure 45. Phaser magnitude response for a one-notch phaser implemented in Simulink.

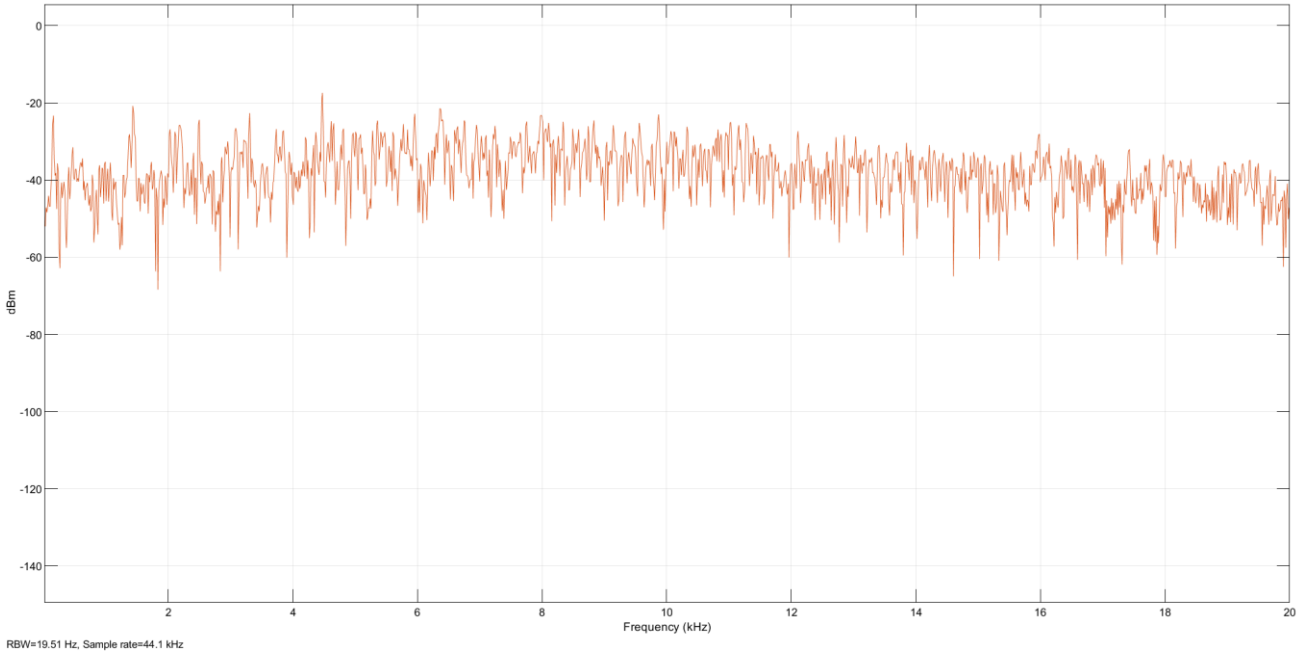

Figure 46. Original spectrum from the white noise used in the phaser implementation.

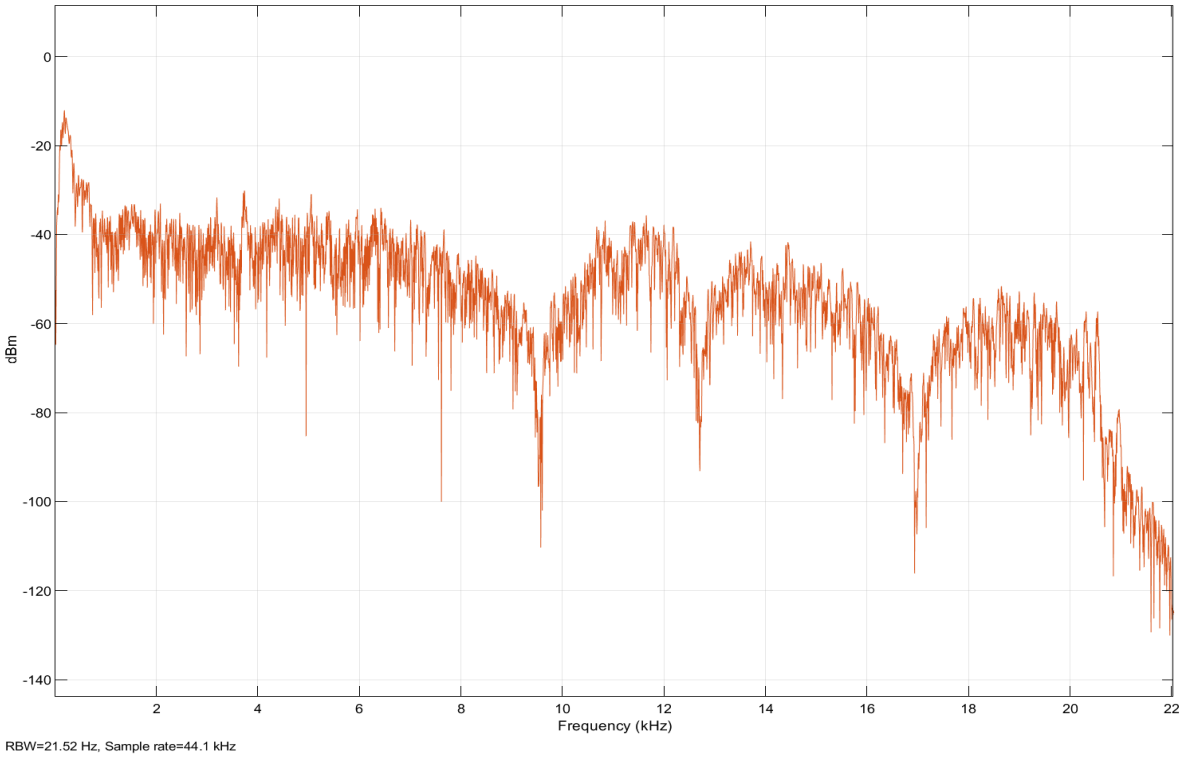

Figure 48. Spectrum captured at the output of the flanger system for FunkyDrums.wav input

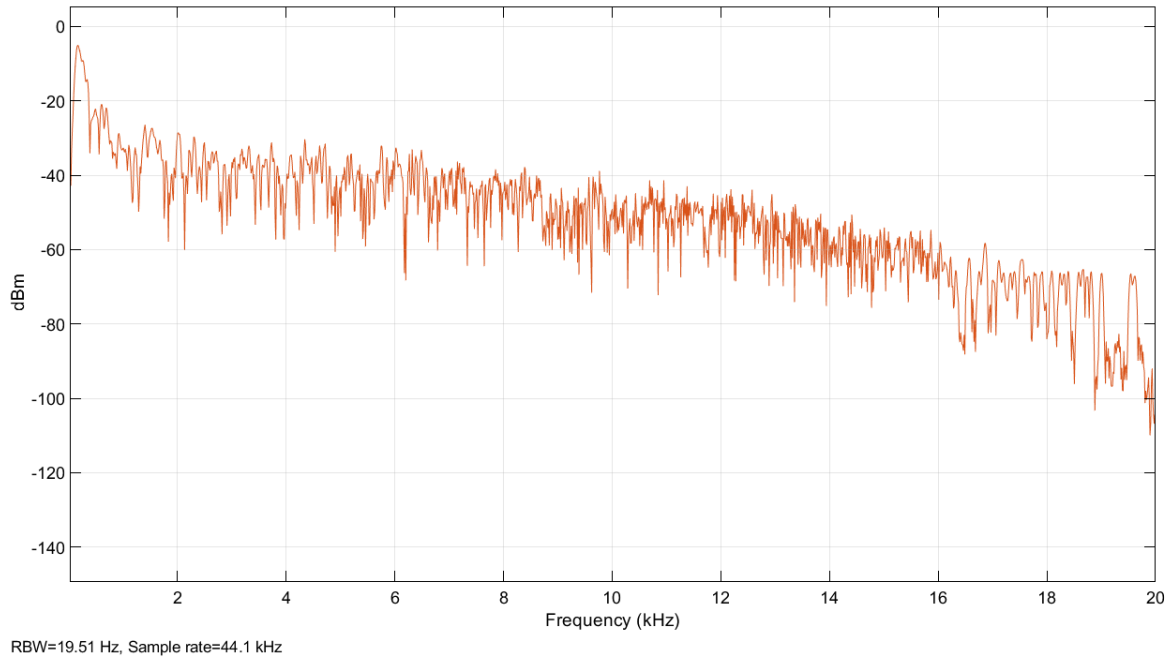

Figure 47. Original spectrum captured for FunkyDrums.wav

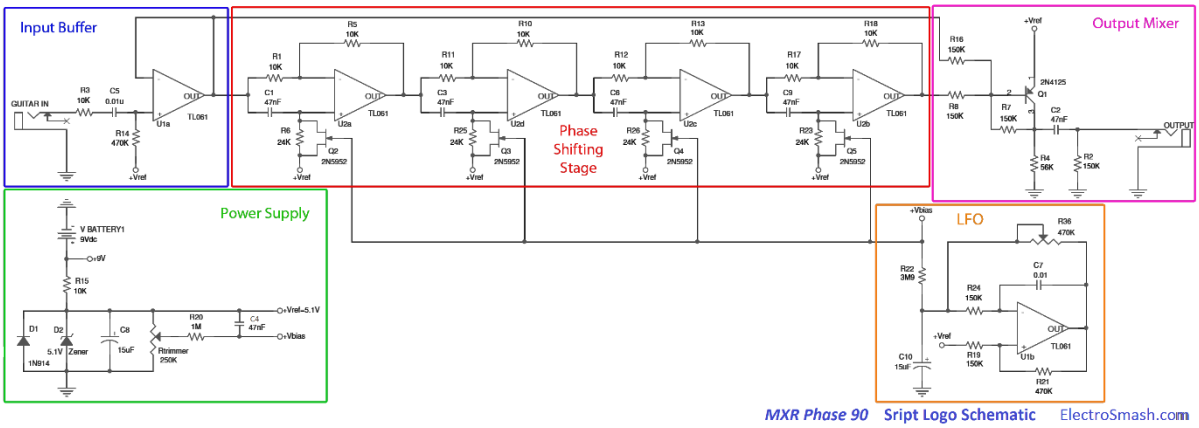

Figure 49. MXR Phaser 90 schematics, obtained from [17].