Analysing and emulating the 2600 envelope and filter modules

By Rasmus Booberg. (Booberg@gmail.com)

Project supervisor: Sten Ternström

As part of the course DT2217

Kungliga Tekniska Högskolan, 2023

Abstract

This project aims to analyse and reproduce the behaviour of the envelope generator and the voltage-controlled filter modules in a Behringer 2600 synthesiser. The first part of the project will focus on analysing, and building understanding, around the signal flow and the response of the modules. This will be done with the goal of identifying unique characteristics that might impact the sound of the synthesiser. The second part of the project will involve reproducing these analog modules digitally within the visual programming language Pure Data. The modelling portion of the project will aim to replicate unique characteristics identified in the analysis of the original analog modules, capturing the nuances of analog circuitry.

The research will primarily utilise voltage measurements to characterise different behaviours, but also include circuit analysis and listening tests. The results of the study should provide insights into the analog behaviour of these modules and the ability of virtual analog modelling to replicate them.

Table of contents

0. Abbreviations

ADSR - Attack, decay, sustain, release
EG - Envelope generator
VCF - Voltage-controlled filter
CV - Control voltage
PD - Pure Data
NTF - nonlinear transfer function

1. Introduction

In 2020 Behringer released their much anticipated 2600 synthesiser, marketed as an “homage” to the renowned Arp 2600, containing supposedly “true-to-the-original analog circuitry”[1]. The “original” ARP 2600 is a classic analog synthesiser that was first introduced in the early 1970s. The ARP 2600 is known for its versatile sound-shaping capabilities and has been used by prominent artists through the decades since its release. Besides applications in music it has also gained fame for producing sounds for the first Star Wars movie, perhaps most notably as the voice of the beloved “droid” R2-D2[2]. How similar Behringers 2600 really is to the original may be up for debate, but this “homage” has certainly piqued a lot of interest across the synthesiser community in the years since it was announced.

The synthesiser consists of various modules that work together to produce a wide range of sounds. Two of the key modules in the Behringer 2600 synthesiser are the attack, decay, sustain, release (ADSR) envelope generator (EG) and the voltage-controlled filter (VCF). These modules are common building blocks in subtractive synthesisers and are used for shaping the sound produced by the oscillators.

The problem addressed by this project is the analysis of the analog behaviour of the ADSR EG and VCF modules in the Behringer 2600 synthesiser and the reproduction of these modules digitally using Pure Data. This project aims to identify the unique characteristics of these analog modules and determine whether they can be accurately replicated digitally. The importance of this problem lies in the fact that analog circuitry is known for its unique and complex behaviour, and there is a growing market for accurate replication of such behaviour. These types of emulations are often called “virtual analog modelling” synthesisers. Understanding the analog behaviour of these modules and the effectiveness of virtual analog modelling could have implications for the design of digital audio equipment and the understanding of analog circuitry.

The first part of the project will focus on analysing, and building understanding, around the signal flow and the response of the modules, with the goal of identifying unique characteristics that might impact the sound of the synthesiser. The second part of the project will involve reproducing these analog modules digitally within the visual programming language Pure Data. The modelling portion of the project will aim to replicate unique characteristics identified in the analysis of the original analog modules, capturing the nuances of analog circuitry.

[1] Behringer | Product | 2600. (Behringer.com 2023)
[2] Sound Design of Star Wars (filmsound.org)

2. Methodology and tools

This chapter lists the central tools and assets that were used during the project.

2.1 Measurements

A Siglent SDS1104x-e oscilloscope was used for all voltage measurements. This is a digital scope with 4 channels, and a frequency limit of 100MHz.

2.2 Implementation

The proposed algorithms were implemented in **Pure Data** (PD). Puredata.info describes the software as “an open source visual programming language for multimedia.”[3]. This was considered a fitting framework as it is a very fast and accessible environment for implementing, testing, and analysing DSP algorithms. Some extensions for Pure Data will be used, the most central to the emulation being the “ELSE” package’s “onepole~” module, a signal controlled, single pole, lowpass filter.

2.3 Schematics

In some cases it was deemed valuable to analyse circuit diagrams of the modules. While the Behringer 2600 schematics are not readily available, the service manual to the original ARP 2600 was available and was used in its place **(see appendix 2 and 3)**. However, as similarity (or lack thereof) between the units could not be confirmed, the overall analysis focused on measurements of the Behringer model where possible.

2.4 Additional software

LTSpice was used to simulate circuits during the research phase of the project.
Matlab was used for generating graphs and for calculating some filter coefficients.

[3] www. PureData.info. 2023.

3. Envelope generator

The ADSR EG is a key module, found in most synthesisers, that is commonly used to shape the amplitude and timbre of the sound over time. The envelope generator receives an incoming signal such as CV trigger and gate, or MIDI notes, and outputs a voltage envelope that can be used as a modulation source.

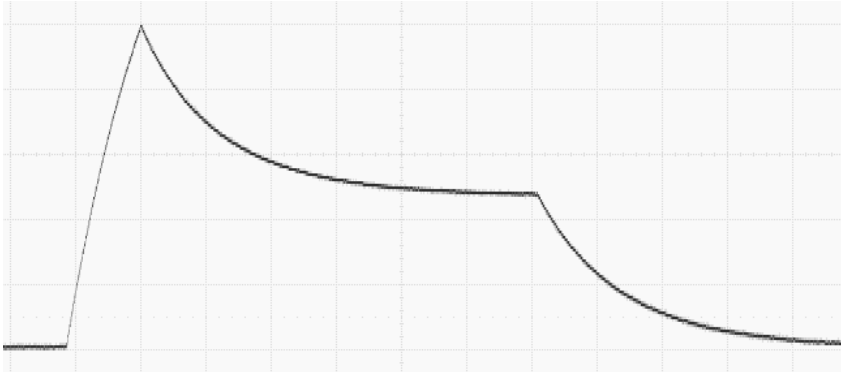


Figure 3.1. CV measured at the output of the behringer 2600s ADSR envelope generator.

An ADSR EG is typically controlled by four parameters: Attack time, Decay time, Sustain level, and Release time. These determine how the envelope cycles through its three phases:

1. Attack - During the attack phase the voltage rises, according to the “attack time” parameter, until it reaches a threshold value at which point the decay stage is triggered.
2. Decay - During the decay phase the voltage drops, according to the “decay time” parameter, to the set “sustain level”. The EG holds at the decay phase as long as it receives a “hold” CV signal. If no such CV is present, the EG passes to the “release” phase.
3. Release - During the Release phase the voltage drops down to zero, according to the “release time” parameter.

The ADSR EG is not, however, required to cycle through these stages fully. In typical use, the EG receives simultaneous “trigger” and “hold” signals, triggering the attack phase. If the “hold” signal would cease during the attack phase, the “release” phase is triggered without first going to the “decay” phase. The EG is susceptible to receiving “trigger” CV while in any state, triggering an attack phase.

3.1 Hypothesis and assumptions

While the basic functionality of ADSR EGs are practically identical across different synthesisers there is one main point where they differ, namely the slope of the transitions between the phases. Analog envelope generators almost exclusively use charging and discharging capacitors to generate these slopes, which leads to the decay and release slopes, given by V_{out} below, following the curve given by the following equation:

$$V_{out} = V_t + (V_s - V_t) \cdot (1 - e^{-t/\tau}) \qquad [3.1]$$

Where V_s is the starting voltage, V_t is the target voltage, and τ decides the decay/release time.

The attack phase works by charging a capacitor towards some reference voltage, triggering the decay phase when the voltage reaches some “decay-trigger” threshold. This gives the following expression for the attack phase:

$$V_{out} = V_{ref} \cdot (1 - e^{-t/\tau}) \quad , \quad for \;\; 0 < V_{out} < V_{dt} \qquad [3.2]$$

Where V_{ref} is the reference target voltage, and V_{dt} is the decay-trigger voltage.

This means that a higher V_{ref}, compared to V_{dt}, gives a straighter curve to the attack envelope.

3.2 Measured slopes

Measurements of the 2600 Synthesizer shows that the ADSR module follows the assumptions presented under **ch3.1**, with a decay-trigger voltage of 10v. By comparing the measured attack envelope with a variety of curves generated using **equation 3.1**, V_{ref} could be approximated to 17.5v, or 1.75 times V_{dt}.

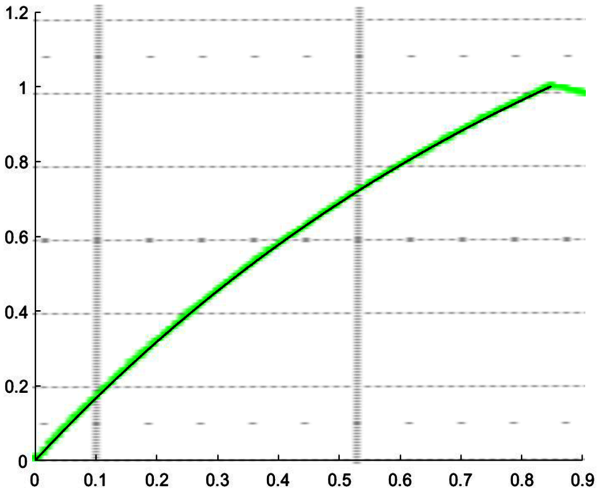


Figure 3.2. A plotted attack curve (black) superimposed over a measured attack curve (green).

The range of the attack, decay, and sustain controls were only measured accurately for their <u>shortest</u> values as only these were seen as significant for the sound character of the synthesiser. The time constant for the decay and release phases were extracted by measuring the time until the signal reached approximately 63.5% ($1 - e^{-1}$) of the target value. For the time constant of the attack phase, first the shortest rise-time was measured as being 0.4ms, then the following equation was solved for τ.

$$\frac{1}{1.75} = 1 - e^{\frac{0.0004}{\tau}} \qquad [3.3]$$

The minimum time constants, τ, measured as the following:

- **Attack** - 0.00047
- **Decay** - 0.0001
- **Release** - 0.00028

3.3 Specific behaviours

To build a rigorous model of the EGs attack and decay phases, a series of tests were performed.

3.3.1 Trigger/Hold CV

By simply sending a “hold” CV to the EG the output would rise according to the decay time parameter, to the chosen sustain level, without going through the ordinary attack/decay stages. The attack parameter did not in any way affect the behaviour of the EG in this case.

In the next test, the EG was triggered “normally”, with simultaneous trigger and hold CV, and the decay parameter was varied to see if this would in any way affect the rise time during the attack phase. This turned out not to be the case, which was considered proof that the attack phase was indeed a wholly separate behaviour from the decay.

Both of these behaviours could be observed by sending CV from the internal sample/hold clock to the EG. This turned out to send the hold CV slightly before the trigger CV, in which case the output rose according to the decay parameter until the trigger CV activated the attack phase.

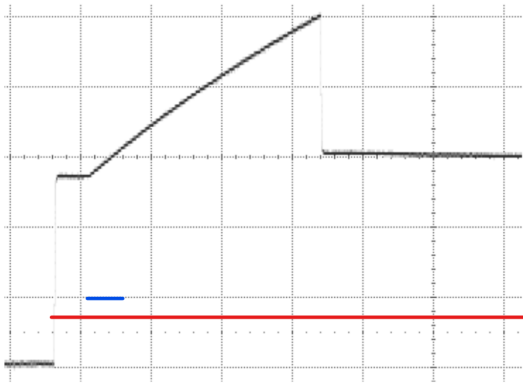


Figure 3.3. Measurement of envelope (depicted in black) when “hold” CV (red) arrives before “trigger” (blue). The first increase in amplitude is triggered by an incoming “hold” CV, with a rise time corresponding to the “decay” parameter. The second (slower ramp) is triggered by an incoming “trigger” CV, with a rise time corresponding to the “attack” parameter.

3.3.2 Decay Trail-off

At particularly short decay times, the decay slope displayed a curious behaviour. The output would drop to about 70% of its intended voltage drop before halting, performing the rest of the voltage drop over a longer period of time. This was particularly noticeable for low sustain levels, where this could cause the audio output to trail out for some time. A similar behaviour was also noticed when only sending hold CV to the module as the voltage would rise to approximately 90% of the intended sustain voltage before halting and slowing down.

3.3.3 Attack/Decay hitchup

During analysis of the fastest possible attack and decay times a quirk was found. The cause of this issue is beyond the scope of this report, however the behaviour can nonetheless be analysed and replicated approximately.

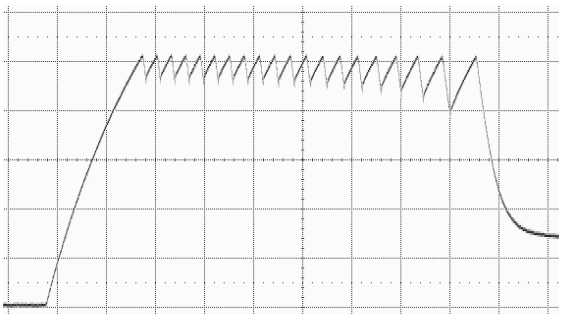


Figure 3.4. Hitchup during quick attack and decay times. All EG parameters set to minimum. Vertical lines represent 200µs, horizontal lines represent 2v.

Triggering the envelope with fast attack and decay times would cause the output voltage to oscillate at its peak before the decay cycle was triggered. Further investigation showed that the attack/decay cycle retriggered rapidly until some time had passed from the time the initial trigger CV was applied. The frequency, amplitude, and the duration (to a lesser extent) of the oscillatory behaviour that emerged was dependent on the attack and release settings. They generally lasted until approximately 2ms had passed from the first trigger and varied in amplitudes within 2v, with a frequency around approximately 16kHz.

3.4 Implementation

The ADSR EG behaviour was implemented as an “external” module for pure data, which basically means that it was implemented as C# code adapted to Pure Data’s data structures. For the sake of brevity, the code itself won’t be explained in much detail in this report, but the “active” part of the code is available in **appendix 1**. Briefly put, the program goes through a series of switch statements according to the state of a “trigger” input, a “hold” input, and the current phase of the EG.

The emulation receives input through 6 different inputs, representing: trigger, hold, attack, decay, sustain, and release. The **trigger** and **hold** inputs receive “audio” type signal buffers and can therefore react in realtime to input. Any value above zero is interpreted as an incoming CV. To replicate the “hitchup” described in **chapter 3.3.3**. The trigger signal should consist of an impulse with a duration of approximately 2ms.

The output for each sample is generated by the following discretisation of the equation for a charging capacitor.

$$y_n = y_{n-1} + adsr \cdot (y_{target} - y_{n-1}) \qquad [3.4]$$

Where the subtext “n” denotes the current sample, “n-1” the previous sample, “target” the target value, and “$adsr$” is a coefficient received as input.

The **attack**, **decay**, and **release** inputs expect pre-calculated coefficients based on the running sample rate, denoted as “SR” below. The user input is expressed as the time constant, “τ”, that were limited at the minimum values previously stated in **ch 3.2**. The coefficients, “adsr” in **equation 3.4,** are calculated through the following equation:

$$adsr = 1 - e^{\frac{-1}{\tau \cdot SR}} \qquad [3.5]$$

Given this method, the various times can be expressed as rise time, in seconds, by the user.

3.4.1 Decay trail-off

The decay tail was unfortunately not implemented in the final model of the ADSR module due to a lack of time for the project. It was planned to be implemented by the addition of a second decay calculation, to be run in parallel with the main one. The first decay calculation would conform to the model described by **equations 3.4 and 3.5**, with the same minimum time constant. But this would only be active over 70% of the intended voltage drop. The second decay calculation would be limited to a higher time constant, relating to the speed of the trail-off, and would span the remaining 30% of the voltage drop. This would mean that the second decay could be neglected for decay times above this, larger time constant, as the result would be identical in theory. The final output would be the sum of the two decay calculations.

3.4.2 Attack/Decay hitchup

At such a high frequency, the oscillatory - sawtooth like, nature of the phenomenon was seen as impractical to implement due to the unpredictable aliasing artefacts that could emerge. This oscillatory behaviour was deemed non-essential to the overall effect on the sound, and that a “good enough” approximation could be achieved by simply sending trigger signals with a 2ms duration into the EG model. This would cause the output to hold at the peak output until 2ms had passed before triggering the decay phase.

This trigger impulse was implemented through the Pure Data module “Step~” (written by Hannes M Zmoelnig), that takes a number, “n_s” representing the number of samples to be output as “1” after triggering. The number of samples corresponding to a 2ms impulse depends on the sample rate, “SR”, and is calculated by the following equation:

$$0.002 \cdot SR = n_s \qquad [3.6]$$

3.5 Conclusion and discussion

The proposed model proved to exhibit a very close approximation to the original hardwares behaviour. As is visible below, in **figure 3.5**, the two are almost identical in some scenarios.

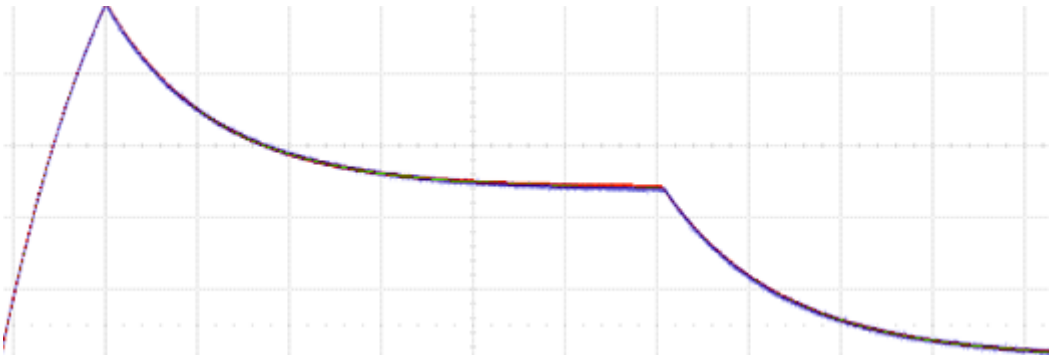


Figure 3.5. A generated ADSR curve, screen captured within Pure Data (red), superimposed over a curve from the Behringer 2600 captured by an oscilloscope (blue).

The model does however deviate slightly for some very short attack and decay times, as was to be expected from the simplified model of the attack “hitch-up” behaviour, and the left out decay “trail-off”. The hitch-up behaviour is considered “close-enough” and is perceived as vital for snappy sounds with quick attack and release times. Future work would involve including the “trail-off” detailed in **chapter 3.3.2**, as this behaviour would probably have a noticeable impact on the sound of the synthesiser in certain scenarios.

No great effort was put into making the response of the controls and accurate representation of the hardware unit, aside from limiting the shortest time constants possible. Nor was accurately representing the control sliders’ range seen as vital to the performance of the emulation at this stage. The control parameters minimum values can determine the character of short, snappy, sounds. This could, arguably, make it an important aspect to emulate accurately. The matter of limiting the parameters at their maximum values was however not seen as holding the same importance.

4. Voltage controlled filter

The filter section of the Behringer 2600 features a resonant 24 dB per octave low-pass filter. The synthesiser actually provides two different filters of which this report focuses on the “4012” variant. This filter is based on ARPs original model 4012 that contains a “transistor ladder”, a topology perhaps most famously found in Moog synthesisers.[4] The filter features an inverting feedback path that can be attenuated by the Resonance slider, a very common feature in synthesisers. High resonance settings cause the filter to go into self oscillation, producing a sine wave at the output, even without the filter receiving any audio input.

Ladder filters, such as the one found in the 2600 synthesiser, have been emulated in a multitude of digital instruments in the past. This part of the project will build on previous documented attempts, adjusting parameters and adding features in an attempt to closely replicate specific behaviours measured on the Behringer 2600.

[4] Analysis of the Moog Transistor Ladder and Derivative Filters. Timothy E. Stinchcombe, 2008. (timstinchcombe.co.uk)

4.1 Hypothesis and assumptions

Resonant ladder filters can, in a broad sense, be described as a series of four single pole low-pass filters in series, with a negative feedback path from the output back into the input.

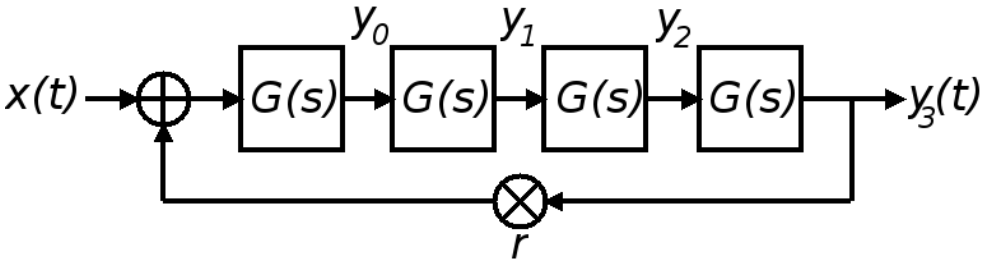


Figure 4.1. Block diagram depicting a general ladder filter topology[5].

This will be used as the base for the suggested implementation, but there are several behaviours to investigate, and implement, beyond this. The main points to investigate going into the project was initially defined as:

- Parameter Response
    - Frequency
    - Resonance
- Clipping
    - Peak to peak range/limits
    - “Hardness/Softness”
    - Nonlinear behaviour in the “linear region”
    - Positions in the signal chain
- Additional features and behaviours
    - Effects of component variance
    - Diminishing resonance oscillation at higher frequencies

The point “effects of component variance” was soon to be admitted as beyond the scope of this project due to the difficulty of measuring such a thing. However, an approximation would be implemented in the model based on standard tolerances in components.

4.2 Frequency Range and CV plotting

To replicate the behaviour of incoming CV, the ARP 2600 schematic diagram was analysed and, in conjunction with some basic assumptions, “accurate enough” behaviours could be detailed. Firstly it’s important to recognise that the filter module is driven by ±15 volts. From this, we can assume that there is a practical limit, at 15v, to the upper range of the filter. By using an external, variable, voltage source it is possible to map the filter's centre frequency as a function of incoming CV.

To determine the cutoff frequency, the filter was set to self-oscillate and the frequency of the resulting sine wave was measured using an oscilloscope. With no incoming CV, and all sliders set at their

[5] Stilson, T.; Smith, J. Analyzing the Moog VCF with considerations for digital implementation. In Proceedings of the 1996 International Computer Music Conference, Hong Kong, China, 19–24 August 1996; pp. 398–401

minimum, the oscillation occurred at about 7.69Hz. This was used as a baseline for 0v. Sending 10v into the CV input, without the lowest attenuation possible set from the front panel, the oscillation occurred at 11.25kHz. By sending 15v, or higher, the frequency maxed out at 154kHz. Notably, the filter is capable of receiving negative CV as well, sending -5v measured the oscillations at 0.238Hz. Lastly, the frequency slider's maximum was measured at approximately 21300Hz with no additional CV present.

Using WolframAlpha to apply a least square fit for exponential functions, these data points could be interpolated to frequency as the following function of CV:

$$F_c = 60.0231e^{0.523332 \cdot CV} - 53 \quad [4.1]$$

And by plugging in the frequency for the maximum slider position and solving for "CV", the slider's maxima could be expressed, in CV, as 11.22v.

4.3 Nonlinear behaviour

The nonlinear behaviour of the filter was first assigned two properties that needed investigating, position and characteristic. The position of the nonlinearities can impact the sound in two ways that were considered significant for this report. The first being position before or after filtering, which can affect at what level any generated harmonics reach the output of the filter. And the second being whether the signal and feedback inputs share headroom or if they saturate in parallel. Furthermore, the characteristics of the nonlinearities were then further divided into two sub-categories, clipping and nonlinearity. The first of these characteristics describes the absolute headroom of the amplifier stages within the filter, and the second describes more subtle nonlinear characteristics that appear across the functioning voltage range. Exceeding the headroom of the amplifier would cause the clipping behaviour, meaning that the outgoing waveform reaches a hard limit at some voltage. This typically generates a lot of harmonic overtones and can be perceived as fairly harsh sounding. Other nonlinear behaviour across the range of the amplifier is more subtle, perhaps even inaudible to the naked ear, generating more subdued harmonics. This second behaviour can however have a big impact on the behaviour of the filter, as will be explained further below. The nonlinear behaviour would be emulated by creating a nonlinear transfer function (NTF) resembling the characteristics observed from measurements.[6][7]

4.3.1 Clipping

Measurements on the synthesiser showed a headroom of approximately ±15v, which is more than enough to avoid clipping the incoming waveforms during standard use. The peaks had just a slight rounding before the absolute maxima, which is to be expected from most analog amplifiers. This behaviour acts to smooth out the sound of the clipping, subduing the generated harmonics. This can have an audible impact as the more "rounded" the clipping is, the less harmonics are generated[8]. If clipping is particularly smooth and rounded, the behaviour is often referred to as "soft-clipping". This

[6] Making digital filters sound analog. Dave Rossum. (umich.edu).

[7] Non-Linear Digital Implementation of the Moog Ladder Filter Huovilainen, A. (bpb-us-w2.wpmucdn.com)

[8] Aliasing Reduction in Soft-Clipping Algorithms. Fabian Esqueda, Vesa Välimäki, Stefan Bilbao. 2015. (eurasip.org)

can be a useful tool in digital sound processing as adding high harmonics can cause unpleasant aliasing, but more on that later.

The “soft-clipping” behaviour at the headrooms limits could be emulated in a number of ways.[9] This was implemented by splicing in a quarter of a sine wave at the ends of a linear function with unity gain, for no other purpose than the perceived simplicity of the method. The following system of equations uses a constant, a, as the fraction representing the “round off” at the limits of the headroom.

$$y_{clip} = \begin{cases} = sin(\frac{x+a}{1-a} \cdot \frac{\pi}{2}) \cdot (1-a) - a & , for \quad -1 < x <- a \\ = x & , for \quad -a < x < a \\ = sin(\frac{x-a}{1-a} \cdot \frac{\pi}{2}) \cdot (1-a) + a & , for \quad a < x < 1 \end{cases} \qquad [4.2]$$

This “round off” was approximated to start at about 94% of the headroom for the analysed filter module.

4.3.2 Nonlinearity

Waveforms within the headroom were not totally linear after passing through the filter stage. A slight curving was visible through various oscilloscope measurements. This phenomenon is common in simple transistor amplifiers and is actually very useful in resonant filters as, apart from adding “character” to the sound, the nonlinear amplification “tames” the feedback and keeps it from growing uncontrollably towards the voltage rails as soon as it reaches a positive open loop gain. In fact, resonant filters that use very clean op-amps for amplification instead often use diodes in various configurations to achieve this type of nonlinear behaviour. Determining a proper NTF for this behaviour was not a straightforward task. Especially as the filter circuit impacts the waveform in other ways, making measurements less reliable. It was decided that it was reasonable enough to assume that the saturation would somehow adhere to a hyperbolic tangent function, as these are commonly used to describe transistors nonlinear behaviour. It was however clear from the measurements that the NFT was nowhere near a pure tanh curve.

Firstly, a transfer curve was captured by passing a sine wave through the filter module and comparing the output to the incoming signal using X/Y metering on the oscilloscope. And secondly, another assessment was made by overlaying a saturated sawtooth wave, measured at the output of the filter module, on top of its “clean” input. The filter's cutoff frequency was set to approximately 150kHz, and the resonance was set at zero, to minimise the impact of the lowpass filtering.

[9] Harmonic Instability of Digital Soft Clipping Algorithms. Sean Enderby and Zlatko Baracskai. Birmingham City University (york.ac.uk)

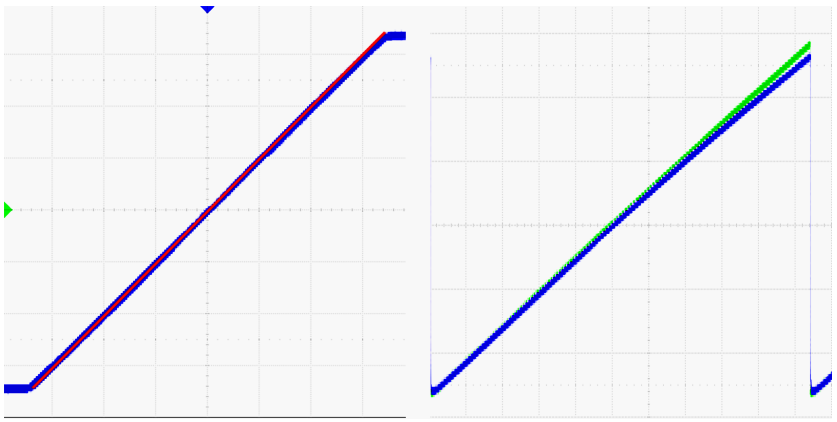


Figure 4.1. Two ways to assess nonlinear behaviour. Left image displays an x/y measurement of input/output (blue), with a straight line (red) superimposed as a reference. The Right image displays input (green) and output (blue) voltages for a purely positive sawtooth wave. Horizontal lines on the rightmost picture represent 2v, vertical lines represent 1ms.

These showed a slant that started to become visible at about ±4v, and the peaks of the waveform were reduced from ±11.12v to approximately ±10.64v. Trials using a pure tanh NTF, with the input and output gain adjusted to match these numbers, did however not yield desirable results for the feedback behaviour. During analysis, the analog filter displayed stable oscillatory feedback at continuously variable levels. The filter would begin to self oscillate at about 70% of the slider's travel length, where the oscillations quickly stabilise at about ±2v. The level of the oscillations increases continuously as the slider is pushed further until it peaks at ±6.3v in amplitude at the slider's 100% position. Using a pure tanh NTF made the self oscillations very unstable, jumping straight from no oscillation to the maximum amplitude with little to no controllability. It was hypothesised that this behaviour could be recreated by attenuating signals over the threshold for self oscillation. And that blending a tanh function with a linear function could be a great method to achieve this.

The positioning of the tanh curve was set so the maxima of the function's second derivative, dubbed the function's "knee", would coincide with the point at which the oscillations appear. This would impart higher levels of attenuation for signals above this point. Adjusting the position of the knee could be done by multiplying the input and output of the function with some constant, dubbed "a" in the equation below, as such:

$$a \cdot y = tanh(a \cdot x) \quad [4.3]$$

To blend this with a linear function, the constant "k" was introduced to represent the fractional amount of the NTF that would be given by the tanh function. The full equation then read as the following:

$$y_{NL} = (1 - k) \cdot x + k \cdot \frac{tanh(x \cdot a)}{a} \quad [4.4]$$

Where "a" adjusts the "knee" of the tanh function and "k" adjusts the "amount" of the tanh function.

4.3.3 Combined expression

Finally, to combine the clipping and nonlinear parts of the transfer characteristics the two equations were combined into the complete NTF as such:

$$y_{NTF} = \begin{cases} = (1-k)[sin(\frac{x+a}{1-a} \cdot \frac{\pi}{2}) \cdot (1-a) - a\,] + k \cdot \frac{tanh(x \cdot a)}{a}, & for \; -1 < x <- a \\ x + k \cdot \frac{tanh(x \cdot a)}{a}, & for \; -a < x < a \\ = (1-k)[sin(\frac{x-a}{1-a} \cdot \frac{\pi}{2}) \cdot (1-a) + a] + k \cdot \frac{tanh(x \cdot a)}{a}, & for \quad a < x < 1 \end{cases} \quad [4.5]$$

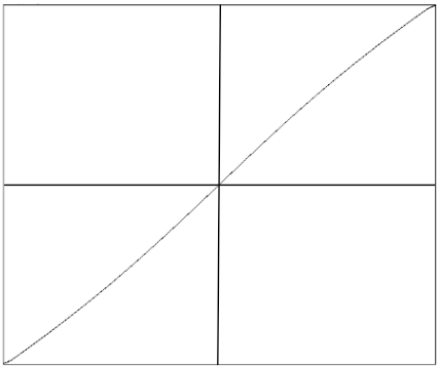


Figure 4.2. A NTF based on **equation 4.5**, generated in pure data.

4.3.4 Position

The positioning of the nonlinear behaviour was seen as significant for two reasons. Firstly, the closer to the output of the filter a nonlinearity appears, the more it should impact the filtered sound. This is because the harmonics would then be attenuated by fewer of the filter stages. Inversely, harmonics that appear from clipping at the input should be fully affected by the filter. The second way saturation positioning was investigated was to determine whether the two inputs shared headroom or should be clipped separately. This was considered important as it could affect the sound of the filter at high resonance values.

The first aspect, positioning towards the input or output, was ultimately not investigated in any great depth, partly due to a lack of resources, and partly due to that listening tests gave no indication that significant nonlinear behaviour was prominent beyond the filter's input stage. This was therefore instead seen as an object for future work and the signal was assumed to saturate at the input.

The question of saturating the input and feedback separately, however, proved to yield significant results. This was investigated after setting up the framework for the emulation by comparing the two alternatives to scope reading of the analog unit, see fig 3.4.

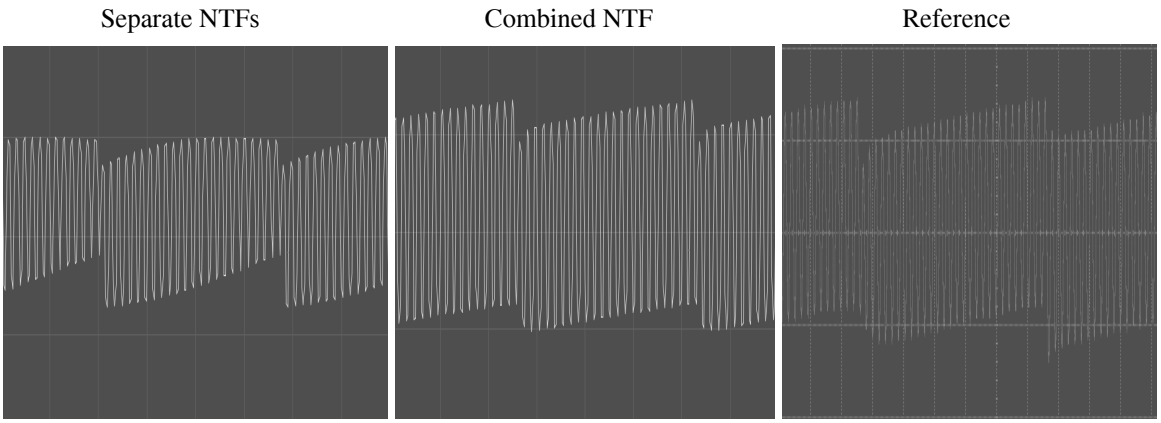


Figure 4.3. Separate NTF for input and feedback (left), summed input and feedback before NTF (middle), measurement from 2600 filter (right).

The comparison (as seen on the images above) was seen as a clear enough indication towards summing the input and feedback before applying the NTF.

4.4 Additional filtering in the feedback path

The schematics for the LFP module and the 4012 filter, shown in **appendix 2 and 3**, shows that the feedback path contains two additional filter steps within the feedback path. One low-pass filter in the output stage of the 4012 filter chip and one high-pass (technically a low-shelving filter but more on that later) filter in the feedback circuit.

4.4.1 Low-pass

The low-pass filter, in the output filter in the 4012 chip, can be summed to 60 picofarad of parallel capacitance directly at the output of a transistor amplifier step. To calculate the cutoff frequency of the filter, the equivalent output impedance of this amplification step must be known. However, taking the values of the capacitors in account it is possible to assess a range of reasonable cutoff frequencies. Using the following equation, the cutoff frequency, Fc, is calculated as a function of the combined capacitor values, 60pF, and the equivalent output impedance, Rekv:

$$F_c = \frac{1}{60 \cdot 10^{-15} \cdot R_{ekv}} \qquad [4.6]$$

This shows that the equivalent output resistance needs to be well into Megaohms for the filter to have any meaningful effect on the phase or frequency response within the range of human hearing. This was seen as a strong enough indication that this filter step could be disregarded in the filter emulation.

4.4.2 Low-shelf

The second filter in the feedback path is technically a low-shelving filter, a high pass path in parallel with a voltage divider. This filter is made up of a 100kΩ potentiometer (the feedback control slider), a

150kOhm resistor in parallel with a 50uF capacitor, and a 2.2kOhm resistor in series. The observed feedback voltage is equal to the voltage drop over the 2.2kOhm resistor. After simulating the circuit in LTSpice it was still unclear if incorporating the behaviour would impact the sound of the final emulation in a meaningful way, as the CF was at most 1.42Hz, but emulating it was deemed a necessary experiment nonetheless. Mostly as the DC behaviour of the filter could, assumingly, be affected significantly.

It was decided that the most precise way of emulating the feedback path would be using a discrete time transfer function. Firstly, the following continuous time transfer function was drawn up symbolically using the Matlab add-on “SCAM”.

$$H_{(s)} = \frac{Pot2 \cdot R4 \cdot (C \cdot R3 \cdot s + 1)}{Pot1 \cdot Pot2 + Pot1 \cdot R3 + Pot1 \cdot R4 + Pot2 \cdot R3 + Pot2 \cdot R4 + C \cdot Pot1 \cdot Pot2 \cdot R3 \cdot s + C \cdot Pot1 \cdot R3 \cdot R4 \cdot s + C \cdot Pot2 \cdot R3 \cdot R4 \cdot s} \quad [4.7]$$

Where Pot1 and Pot2 are the resistance between the potentiometers pins 1-2 and 2-3 respectively. R3 is the 150k resistor, C is the capacitor, and R4 is the 2.2k resistor.

This was converted into discrete time through Euler's forward z-transform, using the generalised sample time “Ts”. After substituting in suitable component values, this could be reduced to four filter coefficients to be applied in the following zero/pole filter:

$$H_{(z)} = \frac{b1 + b2 \cdot z^{-1}}{a1 + a2 \cdot z^{-1}} \quad [4.8]$$

Where:

$$a1 = Pot2 \cdot 16500 \quad [4.9a1]$$

$$a2 = Pot2 \cdot (2200 \cdot Ts - 16500) \quad [4.9a2]$$

$$b1 = 1.65e9 + Pot1 \cdot Pot2 \cdot 7.5 \quad [4.9b1]$$

$$b2 = 1.522e10 - 1.65e9 \cdot Ts + Pot1 \cdot Pot2 \cdot (Ts - 7.5) \quad [4.9b2]$$

4.4.3 Reduced feedback at high frequencies

One unexpected behaviour was discovered while doing measurements on the synthesiser. Having the filter in self-oscillation and sweeping the frequency control showed that the level of the feedback diminishes around 20kHz. It is noticeably reduced as far down as 2kHz, reaches its minimum at around 20kHz, and increases again above this point. No clear reason for this phenomenon could be found from analysing the schematics. It was however hypothesised that the effect could be emulated by adding a bell filter into the feedback path. This could theoretically be tuned to distort the phase response around 20kHz, and thus attenuate the resonance. This was left to be solved during implementation through trial and error.

4.5 Implementation

The implementation of the filter was done through objects already available in pure data. The basic structure is built on cascading four single pole low pass filters, surrounded by a delay loop with a single sample delay.

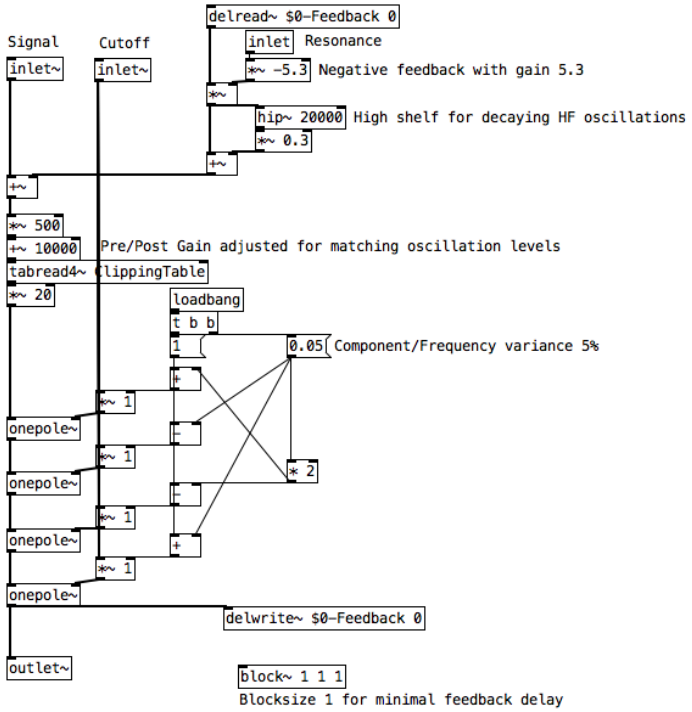


Figure x. The “single sample buffer” section of the filter abstraction.

4.5.1 Upsampling

Upsampling is the process of increasing the sample rate for some digital signal process. This is useful for two reasons in a filter with this topology. One reason being phase cohesion in the feedback path, the second being alias suppression in the nonlinear processing. The upsampling was done through zero padding and filtering. A 10 pole IIR butterworth filter was used, implemented though five cascaded biquad filters. The same filter topology was used as a downsampling filter.

To decide on the amount of oversampling needed, the two factors mentioned above were investigated. The first reason oversampling should be considered is due to the feedback path inherently having one sample delay from appearing at the output to returning back at the input. This delay causes a phase shift according to the sample time, which means that the phase offset increases for higher frequencies according to the following formula:

$$Phase = 360 \cdot f \cdot \frac{1}{SR} \qquad [4.10]$$

Where the phase is measured in degrees, “f” represents frequency, and “SR” sample rate.

To maintain a cohesive phase response it was estimated that the phase shift should not exceed 20 degrees at 20kHz. Plugging this into **equation 4.10** shows that the sample rate should be above 360kHz. To validate using this, as a fitting sample rate, the filter's self-oscillation behaviour was examined as the phase shift needed to be small enough to sustain self oscillation beyond the limit of human hearing. This proved to be the case with 8x oversampling and the emulation running at 48kHz, meaning that the filter would be running at 384kHz sample rate.

The second reason oversampling should be used arises when a signal is passed through nonlinear processing, like the NTFs used in the emulation. This introduces harmonic overtones, which can exceed the Nyquist limit imposed by the sample rate and produce aliasing artefacts. Needless to say, using a higher sample rate therefore reduces the risk of producing these artefacts. With the filter running at 8x oversampling, no audible artefacts were produced from clipping.

4.4.2 Cutoff frequency / CV

Firstly, the range of all incoming control signals had to be made to conform to the CV norm within the synthesiser. This was easily implemented in Pure Data as any controls can be adjusted to any range. The sum of all incoming “CV” was then limited to ±12v, as using settings above this would induce audible aliasing at some settings. The 12v limit still represents a cutoff frequency far enough beyond human hearing to be practically equivalent to the limit of the hardware unit. After clipping, the CV was then converted, according to **equation 4.1**, to represent frequency as the filter objects used this for setting the cut off.

4.4.3 Nonlinearities

A look-up table was used to implement the suggested NTF, a common method used for reducing the computational demand of complex mathematical operations. This method involves solving the NTF for a high number of values, representing the full output range, and then saving the output values to an array. This array is then used to find a corresponding output to any given input. To reduce the size of the array while still keeping rounding errors to a minimum, the look-up process used does a 4 point interpolation for each sample to find a very good approximation of the output.

One instance of the nonlinearity was placed at the filter's input, right after the upsampling filter. The other was placed in the feedback path, right before summing the feedback with the input. The same look-up table (see **figure 4.2**) was used for both nonlinearities.

4.4.4 Additional filtering in feedback path

The proposed feedback circuit was implemented using Pure Data’s “biquad” object. Comparing signals processed through this object with LTSpice simulations showed a perfect match between the two throughout the control range, and across the frequency spectrum. Implementing the filter in the feedback path was however not as successful. Firstly, the outmost positions of the control parameter gave some strange and unwanted behaviour. This was expected as analog components, especially potentiometers, will always impart at least some resistance into the signal path. This unwanted behaviour could be negated by restricting the control parameter to a suiting range. This was sadly not the only issue with the implementation however, the most detrimental being that there were a lot of audible artefacts present, reminiscent of rounding errors. This caused the feedback to impart an “artificial sounding” high end. The method was ultimately omitted. A second implementation was made by summing a high passed signal with an unfiltered one, creating a high shelf filter, and adjusting gain and frequency according to the LTSpice simulations. This ended up “close enough” to the simulations and did not impart the same artefacts as the earlier implementation. The effect was however not deemed noticeable to any reasonable extent and as adding extra multiplications and filtering in such a high sample rate is fairly computationally demanding, the filtering stage was ultimately omitted for the final version of the emulation.

4.4.5 Reduced feedback at high frequencies

The behaviour was already present to some extent in the digital emulation as the sample delay in the feedback path causes increased phase decorrelation towards higher frequencies. However, the attenuation started at a too high frequency and was a bit too steep. The proposed solution, of using a bell shaped equalisation curve, was substituted with a high shelf boost. This was considered more or less equivalent, as the additional high frequency gain would be above the range of human hearing. This solution would also likely be less computationally taxing. Adding a high-pass filter, with a cutoff frequency of 20kHz, in parallel with the existing signal proved to yield a desirable result. With the gain of the high shelf set to +0.3, the behaviour closely matched that of the hardware unit.

4.4.6 Component variance

An experimental "variance" control was implemented that shifted the frequencies of the four filter poles away from the cutoff frequency by an adjustable percentage. The frequency of the poles would be shifted according to the following products of the set variance:

$$f_{c\,p1} = f_c \cdot (1 + 2 \cdot v) \qquad [4.11a]$$

$$f_{c\,p2} = f_c \cdot (1 - v) \qquad [4.11b]$$

$$f_{c\,p3} = f_c \cdot (1 - 2 \cdot v) \qquad [4.11c]$$

$$f_{c\,p4} = f_c \cdot (1 + v) \qquad [4.11d]$$

Where "v" is the adjustable variance, "$f_{c\,pn}$" is the cutoff frequency for pole n, and "f_c" is the cutoff frequency for the full filter.

4.5 Conclusion and commentary

The model was overall considered a great success. Listening tests gave a very convincing impression and after A/B testing the emulation against the hardware it was hard to point out any significant difference. Audio comparisons of the two can be found at **https://youtu.be/ACkoUVlCVkE**.

4.5.1 Upsampling

The IIR based upsampling and downsampling could be said to have been implemented in a rather naive way. However, more effective FIR filters impart delay to the signal, so IIR filters are pretty much the only option for real time audio synthesis. It is perhaps also possible that a "passing" emulation could be implemented using a lower sample rate. The implementation was seen as "good enough" with regards to the scope of the project.

The upsampling itself, resulting in a sample rate of 384kHz seemed necessary. However, this is a fairly high rate for the amount of computations needed for each sample. Using this solution will therefore require a "decent" computer to run, which should be taken into consideration when implementing such an algorithm.

4.5.2 Cutoff frequency/CV

The approach of modelling the CV signals worked fine as an emulation of the hardware's behaviour. However, if this is truly a desirable trait to emulate is another question altogether. A simpler control structure might work just as well and possibly present the user with more precise control. A specific place where emulating CV signals might not be optimal is the trigger and hold functionality. These signals are generated as audio buffers, which is rather computationally taxing compared to more basic signal structures.

4.5.3 Additional filtering in the feedback path

Given more time, the initial shelving filter might have been implemented more successfully. And perhaps this more precise model would impart some character to the sound. This was however considered very unlikely as the simpler, yet functional, approximation that followed the initial failure did not seem to have any noticeable effect on the sound.

4.5.4 Nonlinearity

Even though the suggested NTF was mainly calibrated according to the oscillatory behaviour of the synthesiser, the final NTFs curve did somewhat resemble the measurements (**Fig 4.1**) visually. This might point to the model being a successful emulation in regards to nonlinearities. However, there is still room for improving the NTF in at least one way. The implemented lookup table did in fact not end at a totally "horizontal" curvature, meaning the derivatives at the endpoints were not zero. This means that there is a, even if subtle, hard cutoff to the NTF. Even if the difference might be negligible, making sure that the NTFs endpoints are smoothly rounded off could possibly improve the aliasing behaviour of the emulation.[10]

4.5.5 Reduced feedback at high frequencies

The behaviour was replicated fairly accurately for cutoff frequencies within human hearing. One main difference being that the resonance of the digital emulation continues to trail off at higher frequencies, whereas they only reach a local minima around 20kHz before rising again on the analog synthesiser.

4.5.6 Component variance

The variance imprinted a very subtle change to the filter's sound, even when set beyond reasonable values. But as the downsides of such a feature are miniscule, only four multiplications added to each computation cycle, it was still seen as a success. Using the suggested equations, 4.11a-d, the centre frequency for the filter remained independent of the "variance" parameter. A possible area for future work might be to see how such a variance might interact with nonlinearities in between the filter poles, as this might affect the harmonic content of the output.

[10]Aliasing Reduction in Soft-Clipping Algorithms. Fabian Esqueda, Vesa Välimäki, Stefan Bilbao. 2015. (eurasip.org)

5. References

[1] Behringer | Product | 2600. (Behringer.com 2023)

[2] Sound Design of Star Wars (filmsound.org)

[3] www. PureData.info. 2023.

[4] Analysis of the Moog Transistor Ladder and Derivative Filters. Timothy E. Stinchcombe, 2008. (timstinchcombe.co.uk)

[5] Analyzing the Moog VCF with considerations for digital implementation. Stilson, T.; Smith, J. 1996.

[6] Making digital filters sound analog. Dave Rossum. (umich.edu).

[7] Non-Linear Digital Implementation of the Moog Ladder Filter Huovilainen, A. (bpb-us-w2.wpmucdn.com)

[8], [9] Aliasing Reduction in Soft-Clipping Algorithms. Fabian Esqueda, Vesa Välimäki, Stefan Bilbao. 2015. (eurasip.org)

6. Appendix

Appendix 1.

The following is the code that makes up the logic of the ADSR EG.

```
while(nblock--){
        //Inputs
    t_float Trigger = *in1++;
    t_float Hold = *in2++;
    t_float Attack = *in3++;
    t_float Decay = *in4++;
    t_float Sustain = *in5++;
    t_float Release = *in6++;

        if(Trigger > 0){Phase = 1;}

        switch(Phase){
                case 1:  //Attack
                        Vout = Vprev + (1.75-Vprev)*Attack;
                        if( (Hold == 0) && (Trigger == 0) ){
                                Vout = Vprev - Vprev*Release;                   //1x Release calculation
                                Phase = 4;
                        }
                        else{
                                if(Vout >= 1){
                                        Vout = 1;
                                        Phase = 2;
                                }
                        }
                        break;

                case 2:  //Decay
                        Vout = Vprev + (Sustain-Vprev)*Decay;

                        if(Hold == 0){                                          //Trigger Release
                                Vout = Vprev - Vprev*Decay;                     //1x Release calculation
                                Phase = 4;
                        }
                        else{
                                if(Vout <= Sustain){Vout = Sustain; Phase = 3;}                 //Trigger
Sustain
                        }
                        break;


                case 3:  //Sustain
                        if(Hold == 0){                                                  // Trigger Release
                                Vout = Vprev - Vprev*Release;                   //Discharge to Sustain Value
                                Phase = 4;
                        }
                        else{Vout = Sustain;}
                                                                                        // Rest at sustain value
                        break;

                case 4:  //Release
```

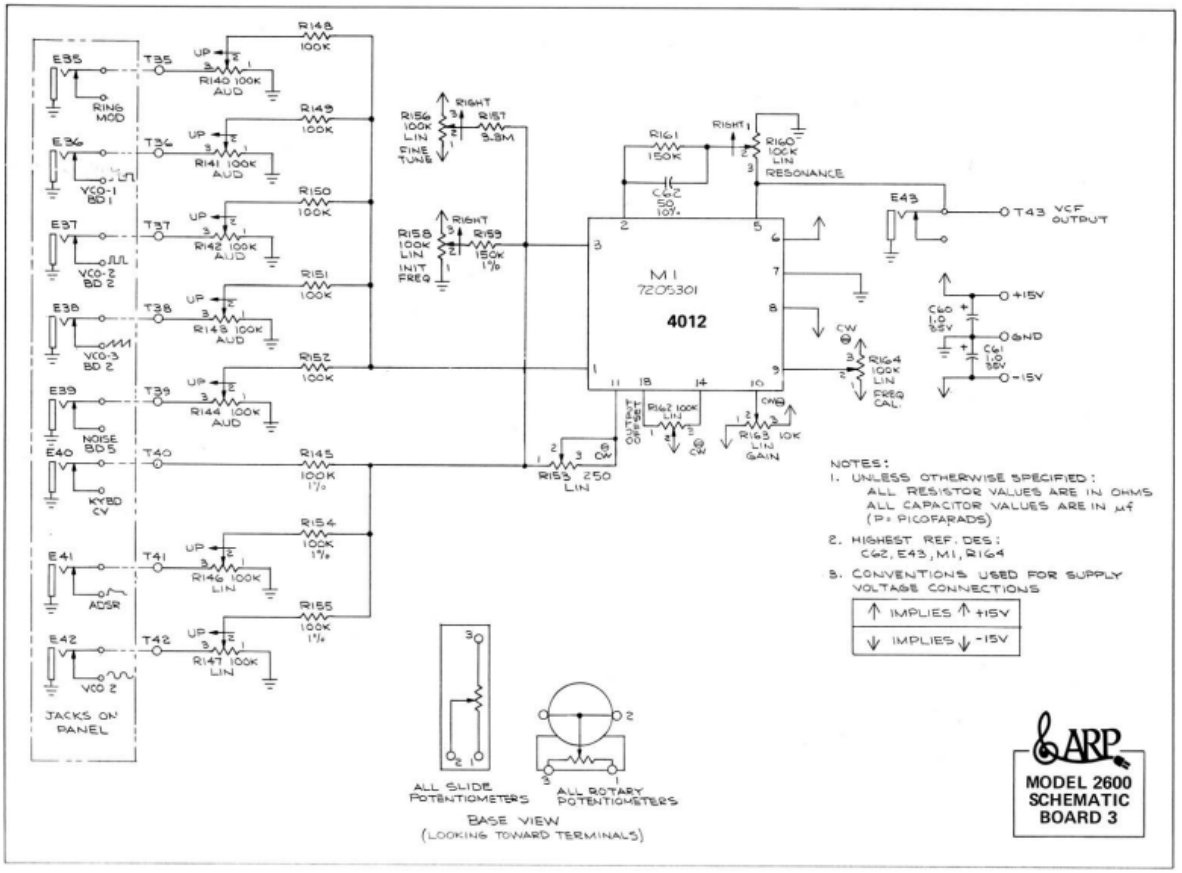
```
                    Vout = Vprev - Vprev*Release;                    //Discharge to 0
                    if(Vout <= 0){Vout = 0; Phase = 0;}
                    break;

            case 0: //Rest                                            // Rest at 0, wait for input
                    Vout = 0;
                    break;
    }


    Vprev = Vout;
    *out++ = Vout;
    };
    x->x_Vprev = Vprev;
    x->x_Phase = Phase;
    return (w + 10);
```

Appendix 2

Schematics for the ARP2600 VCF module.



Appendix 3.

Schematic for the 4012 filter chip, found in the ARP 2600 filter module.