An Analysis of Digital Implementations of the Moog Ladder Filter

Hamish Robb
Oberlin College
hrobb@oberlin.edu

ABSTRACT

The Moog ladder filter is one of the most iconic circuits in analog synthesis. While primarily a lowpass filter, its unconventional design leads to it having a distinctive sonic coloring of its output signal. What sets it apart from other filters used in subtractive synthesis is its transistor based ladder structure, as well as its ability to self-oscillate as a sine wave when its resonance is at its maximum [1]. Many models have been proposed to try to discretize the filter for digital implementation, as well as to construct filters that best reproduce its behavior.

As it turns out, there is no obvious solution for how to construct a digital implementation. All models have tradeoffs, such as in computational complexity, tuning accuracy, Nyquist aliasing, or filter stability. However, in choosing a design, we can decide what qualities are the most important and pick a model that has the best fit.

The goal of this project was to implement many of these models to evaluate their strong suits and weaknesses. 7 models were selected and implemented in Max MSP's gen~ environment. These can be used and modified for further development in creating plugins and applications.

1.INTRODUCTION

Virtual analog modeling is an important topic of DSP research. While there are plenty of analog circuits that are far more complex than the Moog ladder filter, obtaining its characteristic sound is far from trivial. There are many ways of implementing a 4-pole IIR digital filter, but finding ways to emulate how the Moog filter gets its characteristic sound requires in-depth analysis and unconventional techniques.

One common problem when developing a naive model from the filter's difference equations is that most discretization techniques are not feasible [2]. The filter's topology includes a delay-free loop in the digital domain, which is often considered impossible (although, as we shall later observe, these claims of impossibility are rather exaggerated). Another problem is the fact that the Moog ladder filter is nonlinear, which disrupts most discretization techniques and wreaks havoc on the calculation of the filters basic parameters.

2.RELATED WORK

The first major work on the topic of the Moog ladder filter was presented by Stilson and Smith [2]. They analyze the results of many discrete transforms and propose the addition of a unit delay in the feedback loop. While this creates a 5th order linear filter, they identify a way of placing the poles such that the consequences of this are roughly minimized. Wise's work came only shortly later in 1998, proposing a model with the use of an allpass filter [6]. This design is highly unique. While the result we'll show is not ideal as a virtual analog model, the placement of the resonance results in a "bass-boosting" filter. Schmidt's model was self-published in 2009 and is the simplest of these models, to no detriment as we shall observe later [13].

On the topic of linear models, the most promising one has been derived by 3 separate authors. It was first proposed by Fontana in 2007, proving that the zero-delay feedback loop can be resolved mathematically [3]. This results in a linear filter that has the same topology as the analog circuit. Zavalishin also incidentally proposes the same model in his seminal work *The Art of VA Filter Design* [11]. Influenced by Zavalishin and Fontana, Pirkle presents a clear implementation scheme for this model in C++ [12].

The first nonlinear model was presented by Huovilainen in 2004 [8]. A quirk of this model is its "half sample" delay in the feedback loop. The use of tanh functions to clip the signal at various stages makes this model behave more like its analog counterpart, including stable self-oscillation. While this incurs many errors in tuning the filter, this is the root of all nonlinear Moog designs. Huovilainen also proposed a simplified model that optimizes the number of tanh functions per sample [8]. D'Angelo and Valimaki improve upon this with a modified placement of the nonlinearities as well as a set of tuning equations [7].

Puckette's model was originally written for Pure Data as the bob~ external [10]. Built from the works of Stilson, Smith, Huovilainen, and Stinchcombe [4], this performs a direct Runge-Kutta integration of the differential equations of the circuit.

3.DESIGN

The core Max patch of this project was designed to be able to present and platform each model in a roughly unified way. This had many implications towards the implementation of each algorithm, which shall be described later.

3.1.Normalization of Parameters

The resonance of each filter was implemented to scale roughly between no resonance (0) and self-oscillation (1). The frequency range was chosen to be between 0 Hz and 22050 Hz. Every subpatch was designed to implement these few controls, which meant sacrificing some low level features such as overdrive, clipping factor, and volume control. Attempts were made in implementation to ensure that these filters were stable and could be roughly compared to each other.

3.2.Volume Control

It is impossible to completely normalize these filter algorithms to be equivalent. While tuning tables and more advanced volume correction could have been used, what a resonance of 1.0 means to one filter could mean something completely different to another. The nonlinear components of these models can sometimes cause unpredictable results in volume scaling and even frequency control. It was decided to normalize the input by the use of a soft clipper, as well as the output. Since these methods are not absolute, a peak limiter was placed on the output to ensure minimal harmonic distortion beyond what the filter does internally. While these attempts to control these filter are quite imperfect, they lay the groundwork for more refined implementations in the future.

3.3. Aliasing

Not all of these models are nonlinear, but many of them have quirks that require special mitigation to achieve the correct frequency response and also to deal with Nyquist aliasing. Each filter was built in a subpatch that could be loaded by Max's poly~ so that poly~'s oversampling could be used. It was decided to run each filter at 16x oversampling based on an operating sample rate of 44.1khz. This is not ideal for any actual implementation, but gives each filter enough room to operate as ideally as possible.

A biquad lowpass filter was placed on the output of each filter with a cutoff below 20khz. While this may affect the high frequency response of some filters, it was decided that this was not consequential because these filters are already operating to reduce high frequencies in the first place.

4.IMPLEMENTATION

The implementation of these filters were based on a combination of their original block diagrams, difference equations, source code, or derivative code within reason based on the original papers. This next section will outline the specific choices and challenges encountered in implementing each filter in gen~.

4.1. D'Angelo & Valimaki

Two models were built in gen~ from this paper. The first is based entirely on D'Angelo's Octave code. It was decided that this code was idiomatic in a way that a separate implementation taking a more direct translation from the original block diagrams was necessary. The original paper has some mistyped equations which are resolved in the errata linked to the paper. While the authors cite the voltage of a transistor at room temperature (Vt) as 26mv, it is unclear how it should be scaled in a discrete digital context. Ultimately, Vt was implemented to be 0.026.

The scaling caused by the small value of Vt meant that the output was exceedingly quiet. Attempts were made to adjust this, but ultimately it was easier to perform these corrections on the output and input rather than internally to preserve the intended qualities of this model. However, there was not a clear way to scale the resonance into a value between 0 and 1. It is dependent on the input volume, so while a strong sine wave may occur with near-silent signals, the same resonance may be imperceptible on aloud signal.

4.2. Fontana, Pirkle, & Zavalishin [3][11][12]

This implementation is a direct translation of Pirkle's C++ code. This was selected because of its clarity. A challenge in implementing this in gen~ is the amount of fine-tuned wiring necessary to have a clear signal flow when graphically programming. Ultimately, it was designed for typed code and was implemented as such.

4.3. Huovilainen [5]

The same problem as D'Angelo and Valimaki's model in regards to the temperature of a transistor was encountered. One implementation of this filter defined this variable as 40000, while another defined it as 26. In this case, it was set to 0.026. Some tuning equations were found online, either sourced by Victor Lazzarini or Huovilainen. This was originally from the CSound opcode for MoogLadder, although the opcode could not be found [14]. The relationship between the tuning and the transistor temperature variable is unknown.

4.4. Puckette [10]

This implementation used Puckette's code directly with some simplifications made. Most notably, the variable clipping function was discarded in favor of a direct tanh nonlinearity (which is equivalent to running Puckette's model with a factor of 1).

4.5. Schmidt [13]

Schmidt's model was fairly easy to implement. The most difficult part was implementing his proposed passband gain correction. Notably, a Q factor was not selected for the low-shelf filter. Max's filtergraph~ was used to implement this instead.

4.6. Stilson and Smith [2]

The implementation of this filter is based on the diagrams and equations proposed by Valimaki and Huovilainen in 2006 [8]. The most important part of this model is getting the desired pole placement as shown by Stilson and Smith's "compromise" model. The resonance had to be limited to ensure that this filter was stable under most conditions.

4.7. Wise [6]

This paper did not include a tuning equation, so one was calculated based on Wise's equations. The tuning error for Wise's implementation is significant, and appears to be between 1/4 and 1/2 of the desired resonant frequency. Some attempts were made to mitigate this, but an analysis of the circuit is necessary to determine if this is a problem with the allpass configuration or the equations. It was confirmed that the derived equations properly scales the coefficient 'a' between -1 and 1.

$$\omega_c = \frac{2\pi f_c}{f_s}$$

$$a = -\frac{|\sin \omega_c| - 1}{\cos \omega_c}$$

5.APPLICATION/EVALUATION

The problem with analyzing the results of each filter is that there aren't easily calculable methods to analyze nonlinearities in digital filters. Finding the amplitude and phase response of each model is enough for 7 separate research papers. A graphic and qualitative method is used instead. Each filter was given a linear frequency sweep over white noise. Variables used for these recordings were resonance (0 or 1) and input gain (0.25, 1, 4). In addition, self-oscillation tests were performed by putting the input gain at 0.01 and resonance at 1. In addition, the same process was used on a musical example to demonstrate the qualities of the filter. All of these recordings can be found in the audio subfolder.

These recordings were plotted into spectrograms for a visual component. These spectrograms are the most useful for visualizing the location of the resonant peak over a linear sweep. While this does not accurately describe the filters cutoff behavior, this is useful to ballpark how accurate the tuning for each filter is.

There were difficulties getting clean plots to include in this paper, however we shall note the takeaways that can be observed from both the 0-resonance 1-gain and 1-resonance 0.001-gain spectral images.

The most unique response is on Wise's model, which does not self-oscillate and appears to have a resonant peak placement which is extremely inaccurate [6]. It should be noted that this error's result is a highly unique filter which will be used for future musical endeavors.

The nonlinear filters all show some error, especially in D'Angelo and Valimaki's [7]. While at certain volume levels these nonlinearities can be evaluated as linear, this is technically incorrect above a certain threshold. However, attaining an ideal response in a nonlinear discrete system may be unrealizable and also inaccurate to how the analog filter functions.

The linear filters unsurprisingly have the most accurate responses, but the worst self-oscillation. Most of them are silent or unstable. The best performing self-oscillating filter is Stilson and Smith's, which is blaring and harsh. Nonlinearities are necessary to prevent and control this [2].

The other problem with analyzing these filters is that it is hard to quantify descriptors such as "warmth", distortion, or just how accurate they are to our mental conception of what a Moog filter should sound like.

6.FUTURE WORK

A more refined implementation of the Moog ladder filter can be derived from the results of this project. Implementing tuning tables is a good first step in taking this further. Another would be translating this code to a library such as JUCE for wider distribution.

Another proposal is to combine some of the features of these filters to see if they may bring out better qualities. Using more strategic nonlinearities in the linear filters will allow them to self-oscillate and sound more like the idealized filter.

An analysis of computation complexity is also required. As stated earlier, these filters were run at 16x oversampling. This is unrealistic, and it should be noted that the tanh nonlinearity is quite expensive to have even just once in a filter. The average CPU usage while running these filters is between 20 and 30% on a 2020 ARM M1 MacBook Pro.

There are many matrix-based implementations that have been proposed. These were not examined because they are often optimizations of existing models, and additionally Max's gen~ does not have any data structures that could easily be used to implement these. However, this highlights that an external package with matrix operations for gen~ could be useful.

One last proposal for future work is the use of machine learning to optimize and also make it easier to find ideal tunings for these filters. This may reduce the amount of work and computation necessary to find best fit cubic equations for a better frequency response. While complete accuracy is not true to life, it should be noted that sometimes the Moog filter is used as a VCO in self-oscillation. Finding a stable tuning system that allows for that is ideal.

7.CONCLUSION

The Moog ladder filter is deceptively hard to discretize. Between finding the right topology and being able to optimize the performance of the nonlinearities, there are too many variables to quantify at once for an efficient analysis. Even the most on-paper perfect models may lack qualities considered essential for the Moog filter.

While D'Angelo and Valimaki's model is the most recent and holds the most promise, its frequency response is currently undesirable and in need of further refinement. The best model implemented is clearly Huovilainen's [7] [5]. It has the best frequency response, has an ideal distorted sound when pushed, and self-oscillates within bounds. It should be noted that this model cannot sustain self-oscillation under certain conditions, as noted by D'Angelo and Valimaki.

8.REFERENCES

[1] Moog, Robert A.. “A Voltage-Controlled Low-Pass High-Pass Filter for Audio Signal Processing.” Journal of The Audio Engineering Society (1965): n. pag.

[2] Stilson, Timothy S. and Julius Orion Smith. “Analyzing the Moog VCF with Considerations for Digital Implementation.” (1996).

[3] Fontana, Federico. “Preserving the Structure of the Moog VCF in the Digital Domain.” ICMC (2007).

[4] Stinchcombe, Timothy “Analysis of the Moog Transistor Ladder and Derivative Filters” https://www.timstinchcombe.co.uk/synth/Moog_ladder_tf.pdf (2008).

[5] Huovilainen, Antti. “Non-linear digital implementation of the Moog ladder filter” (2004).

[6] Wise, D. K.. “The Recursive Allpass as a Resonance Filter.” ICMC (1998).

[7] D’Angelo, Stefano and Vesa Välimäki. “An improved virtual analog model of the Moog ladder filter.” 2013 IEEE International Conference on Acoustics, Speech and Signal Processing (2013): 729-733.

[8] Välimäki, Vesa and Antti Huovilainen. “Oscillator and Filter Algorithms for Virtual Analog Synthesis.” Computer Music Journal 30 (2006): 19-31.

[9] Smith, Julius Orion and Timothy S. Stilson. “Efficiently-variable non-oversampled algorithms in virtual-analog music synthesis: a root-locus perspective.” (2006).

[10] Puckette, Miller. Bob~ source code. https://github.com/dmedine/bob--windows (2015)

[11] Zavalishin, Vadim. “The Art of VA Filter Design” (rev 2.1.0, 2018) https://www.native-instruments.com/fileadmin/ni_media/downloads/pdf/VAFilterDesign_2.1.0.pdf

[12] Pirkle, Will “Virtual Analog (VA) Filter Implementation and Comparisons v2.0” (2013)

[13] Schmidt, Robin “Resonance Tuning for the digital Moog Filter” http://www.rs-met.com/documents/dsp/ResonanceTuningForTheDigitalMoogFilter.pdf (2009)

[14] Budde, Christian. “Moog Filter”. https://www.musicdsp.org/en/latest/Filters/196-moog-filter.html (2005)