A COMPARISON OF VIRTUAL ANALOG MODELLING TECHNIQUES FOR DESKTOP AND EMBEDDED IMPLEMENTATIONS

Jatin Chowdhury

Center for Computer Research in Music and Acoustics
Stanford University
Palo Alto, CA
jatin@ccrma.stanford.edu

ABSTRACT

In this writing, we develop a virtual analog model of the Klon Centaur guitar pedal circuit, comparing various circuit modelling techniques. The techniques analyzed include traditional modelling techniques such as nodal analysis and Wave Digital Filters, as well a machine-learning technique using recurrent neural networks. We examine these techniques in the contexts of two use cases: an audio "plug-in" designed to be run on a consumer-grade desktop computer, and a guitar pedal-style effect running on an embedded device. Finally, we discuss the advantages and disdvantages of each technique for modelling different circuits, and targetting different platforms.

1. INTRODUCTION

The Klon Centaur is an overdrive guitar pedal designed by Bill Finnegan in the early 1990's, that has developed cult acclaim amongst guitarists [1]. The circuit is notable for producing "transparent distortion" [2], a term used to describe the way the pedal seems to add distortion to a guitar's sound without otherwise affecting the tone. While the original manufacturing run of the pedal ended in 2004, many "clones" of the pedal have been produced by other manufacturers, adding to its cult following.

Circuit modelling is typically broken down into "white-box" and "black-box" approaches [3]. A "white-box" approach uses knowledge of the internal mechanisms of the circuit, often modelling the physical interactions of the electrical components. Popular white-box methods include nodal analysis [4], Port-Hamiltonian analysis [5], Wave Digital Filters [6, 7], and nonlinear state space analysis [8].

"Black-box" circuit modelling methods generally use measurements taken from the circuit being modelled and attempt to model the response of the circuit without knowledge of the internal workings of the system. Traditional black-box techniques include impulse response measurements [9] and extensions thereof, including the Weiner-Hammerstein method [3]. Recently, researchers have begun using machine learning methods for black-box modelling. In [10], the authors use a WaveNet style architecture to generate an output signal sample-by-sample. In [11] the authors use a deep fully-connected networks to approximate nonlinear state-space solutions for a circuit, effectively a "grey-box" approach. Finally, in [12] the authors use a recurrent neural network to model the behavior of audio circuits with control parameters.

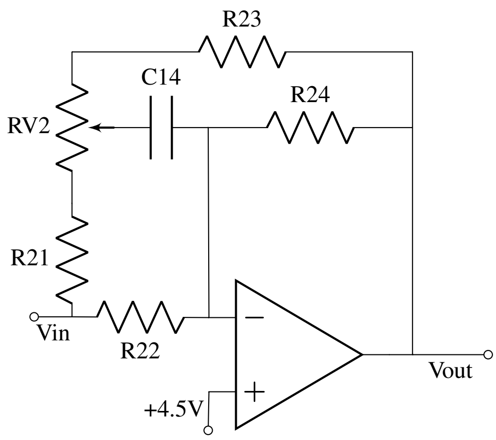

Figure 1: *Klon Centaur Tone Control Circuit*

2. TRADITIONAL CIRCUIT MODELLING TECHNIQUES

First, we examine the use of traditional circuit modelling techniques, specifically nodal analysis and Wave Digital filters, using sub-circuits from the Klon Centaur as examples.

2.1. Nodal Analysis

The process for creating a digital model of a circuit using nodal analysis is as follows:

1. Convert the circuit into the Laplace domain.
2. Form a Laplace domain transfer function of the circuit.
3. Use a conformal map to transform the circuit into the digital domain.

As an example circuit, we examine the Tone Control circuit from the Klon Centaur (see fig. 1). The first step is to convert the circuit into the Laplace domain, using the Laplace variable $s = j\omega$. The impedances for each principle circuit component: resistors (Z_R), capacitors (Z_C), and inductors (Z_L), are as follows:

$$Z_R = R, \quad Z_C = \frac{1}{Cs}, \quad Z_L = Ls \tag{1}$$

From there, using linear circuit theory, one can construct a Laplace Domain transfer function for the circuit. Note that this assumes an ideal operational amplifier operating in its linear region. For more

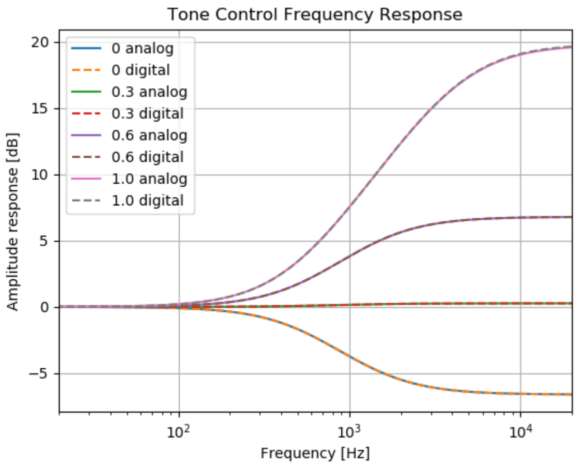

Figure 2: *Tone control frequency response at various values of the Treble parameter, comparing the responses of the analog filter with the digital model.*

information on this process, see [13]. For the tone control circuit, the Laplace domain transfer function can be written as:

$$\frac{V_{out}(s)}{V_{in}(s)} = \frac{C_{14}\left(\frac{1}{R_{22}} + \frac{1}{R_{21}+R_{v2b}}\right)s + \frac{1}{R_{22}}\left(\frac{1}{R_{21}+R_{v2b}} + \frac{1}{R_{23}+R_{v2a}}\right)}{C_{14}\left(\frac{1}{R_{23}+R_{v2a}} + \frac{1}{R_{24}}\right)s + \frac{-1}{R_{24}}\left(\frac{1}{R_{21}+R_{v2b}} + \frac{1}{R_{23}+R_{v2a}}\right)} \tag{2}$$

Note that we refer to the the section of potentiometer R_{v2} that is above the wiper as R_{v2a}, and the section below as R_{v2b}, and that we ignore the DC offset created by the 4.5V voltage source at the positive terminal of the op-amp.

Next we use a conformal map to transform the transfer function from the Laplace domain to the z-plane where it can be implemented as a digital filter. The most commonly used conformal map is the bilinear transform, defined as

$$s \leftarrow \frac{2}{T}\frac{1-z^{-1}}{1+z^{-1}} \tag{3}$$

Where T is the sample period of the digital system. For more information on the use of the bilinear transform to digitize an analog system, see [14]. The resulting filter is known as a “high-shelf” filter, that accentuates high frequency content in the signal. The resulting frequency response of the digital model, validated against the response of the analog circuit is shown in fig. 2.

2.1.1. Advantages and Limitations

The advantages of nodal analysis are that the circuit model is simple and computationally efficient. The model can be constructed with minimal knowledge of simple circuit theory, and basic understanding of digitial signal processing. The main disdvantage is that nodal analysis cannot be used for nonlinear circuits, though it can be extended to model this class of circuits through modified nodal analysis (MNA) [15]. Another disadvantage of nodal analysis-based methods is that (typically) large portions of the system need to be recomputed when a circuit element is changed, such as a potentiometer. While this computation is fairly simple in the

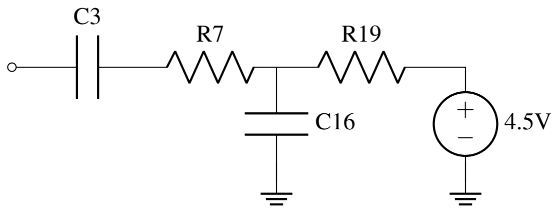

Figure 3: *Klon Centaur Feed-Forward Network 1 Circuit*

Figure 4: *WDF tree for the Klon Centaur Feed-Forward Network 1 Circuit.*

example shown here, it can become vastly more difficult for more complex systems.

2.2. Wave Digital Filters

The Wave Digital Filter (WDF) formalism allows circuits to be modelled in modular and flexible manner. Originally developed by Alfred Fettweis in the 1970’s [6], WDFs have recently gained popularity in modelling audio circuits, and have been extended to model a wider class of circuits [7]. The WDF formalism defines each circuit element as a port with some characteristic resistance R_0, and uses wave variables passing through each port, rather than the typical voltage and current variables. The incident wave at a certain port is defined as:

$$a = v + R_0 i \tag{4}$$

where v is the voltage across the port, and i is the current passing through the port. The reflected wave is similarly defined as:

$$b = v - R_0 i \tag{5}$$

A Wave Digital Filter defines circuit elements (resistors, capacitors, inductors, etc.) in the wave domain, and allows the elements to be connected by series and parallel adaptors also defined in the wave domain. The full derivation of these WDF elements is given in [6] and [7].

Once each circuit element and adaptor has been defined, they are connected together in a strcuture often referred to as a WDF tree. As an example, we examine the “feed-forward network 1” from the Klon Centaur circuit (see fig. 3).

The corresponding WDF tree is given below. @TODO... Also shwo simulation results. @TODO...

2.2.1. Advantages and Limitations

The primary advantage of the Wave Digital approach is its modularity. The ability to construct a circuit model with each circuit component treated completely independently in the digital domain opens up many interesting possibilities for circuit prototyping, modelling circuit-bent instruments and more. Additionally, this modularity allows each circuit component to be discretized separately, even using different conformal maps, which can improve model behavior for certain classes of circuits (see [3]). Finally, the separability of components means than when a component is changed (e.g. a potentiometer), the component change is propagated so that only components with behavior that depends on the impedance of the changed component need to be recomputed.

The main disdvantage of WDFs is its difficulty in handling circuits with complex topologies, or multiple nonlinearities. While the recent addition of $\mathcal{R}$-type adaptors to the Wave Digital formalism [7] has begun to make these circuits tractable, the WDF models of these types of circuits are significantly more computationally complex, and somewhat compromise the modularity of the system.

3. RECURRENT NEURAL NETWORK MODEL

While several styles of machine-learning based models are available for modelling analog audio circuitry [10, 11, 16], we choose the recurrent neural network approach developed in [12] as our starting point. Using a recurrent neural network (RNN) allows the for a significantly smaller neural network than would be possible with a traditional deep neural network or convolutional neural network, meaning that the network can be evaluated much faster for real-time use, while maintaining a smaller memory footprint (an important advantage on embedded platforms). Additionally, recurrent neural networks are a sensible candidate for modelling distortion circuits, particularly circuits with stateful behavior, given the fact that recurrent network building blocks, such as gated recurrent units, themselves resemble audio distortion effects, and can directly be used as such [17]. Finally, the model proposed in [12] allows for the inclusion of control parameters into the model, such as potentiometer positions, helping to solve a persistent limitation of black-box style circuit models.

In the following paragraphs, we outline the use of an RNN for modelling the gain stage circuit from the Klon Centaur pedal. While our model is similar to the model used in [12], it differs in some notable ways. For instance, the model described in [12] accepts the values of control parameters to the circuit as inputs to the RNN, however we were unable to successfully train a network in this fashion. Instead, we construct separate networks for five different values of the "Gain" parameter, and fade between the outputs of the networks in real-time in the final implementation of the model. Other differences are outlined further below.

3.1. Model Architecture

The model architecture described in [12] consists of a single recurrent layer followed by a fully connected layer consisting of a single "neuron" (see fig. 5). In our models, we use a recurrent layer made up of 8 Gated Recurrent Units. For training, all models are implemented in `Python` using the `Keras` framework [18].

3.1.1. Recurrent Layer

Recurrent layers are typically comprised of one of two types of recurrent units: Long Short-Term Memory units (LSTMs) or Gated Recurrent Units (GRUs). For this application, we choose to use GRUs [19] since they require fewer operations, allowing for faster computation, and since they requre fewer weights, thereby allowing the model to have a smaller memory footprint. The GRU consists of three "gates": the update gate $z[n]$, reset gate $r[n]$, and the new gate $c[n]$. These gates are used to compute the cell's current output $h[n]$ from its current input $x[n]$ and previous output $h[n-1]$ as follows:

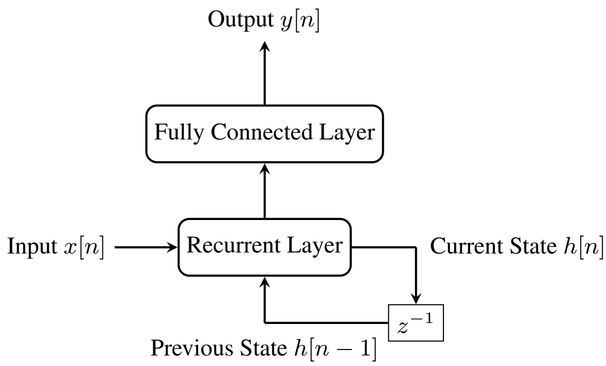

Figure 5: *RNN Architecture.*

$$z[n] = \sigma(W_z x[n] + U_z h[n-1] + b_z) \quad (6)$$

$$r[n] = \sigma(W_r x[n] + U_r h[n-1] + b_r) \quad (7)$$

$$c[n] = \tanh(W_c x[n] + r[n] \circ U_c h[n-1] + b_c) \quad (8)$$

$$h[n] = z[n] \circ h[n-1] + (1 - z[n]) \circ c[n] \quad (9)$$

Where W_z, W_r, W_c are the kernel weights for each gate, U_z, U_r, U_c are the recurrent weights for each gate, and b_z, b_r, b_c are the biases for each gate. Note that as the inputs and outputs to the GRU layer may be vectors, all products in the above equations are assumed to be standard matrix-vector products, except those Hadamard products denoted $\circ$.

3.1.2. Fully Connected Layer

A fully connected layer computes an output vector $y[n]$ from input vector $x[n]$ as follows:

$$y[n] = \alpha(W x[n] + b) \quad (10)$$

Where W is the kernel weights, b is the layer bias, and $\alpha(x)$ is the layer activation. In our model, we use no activation, i.e., $\alpha(x) = x$.

3.2. Training Data

Our dataset consists of 4 minutes of electric guitar recordings, from a variety of electric guitars including a Fender Stratocaster and a Gibson Les Paul. The guitars are recorded "direct" meaning that the recorded signal is equivalent to the signal received by the pedal coming directly from the guitar. Recordings were made using a Focusrite Scarlett audio interface at 44.1 kHz. Note that this sample rate is very low compared to that used for other neural network models of nonlinear audio effects [10, 12], however, this sample rate was chosen because the embedded hardware on which the final model was implemented processes audio at this sample rate. The recordings were then separated into segments of 0.5 seconds each, resulting in a total of 425 segments.

Since the original Klon Centaur is quite expensive, we used a

SPICE simulation of the Centaur circuit in order to obtain a "ground truth" reference dataset. The reference dataset measures the output voltage of the summing amplifier from the circuit at five different values for the "Gain" potentiometer.

3.3. Training

We trained our models on 400 of the 425 audio samples, saving 25 samples for validation. Training was performed using the Adam optimizer [20], with an initial learning rate of 2×10^{-3}.

4. IMPLEMENTATION

In order to compare the virtual analog methods described above, we construct two emulations of the Klon Centaur circuit: one emulation using traditional circuit modelling methods, and a second using a recurrent neural network. The Centaur circuit can be broken down into four separable parts:

1. Input Buffer
2. Gain Stage
3. Tone Control
4. Output Buffer

Due to their simplicity and linearity, in both emulations the input buffer, output buffer, and tone control circuits were modelled using nodal analysis. The "Gain Stage" circuit can be further broken down into six (mostly) separable parts:

1. Feed-Forward Network 1 (FF-1)
2. Feed-Forward Network 2 (FF-2)
3. Pre-Amp Stage
4. Amplifier Stage
5. Clipping Stage
6. Summing Amplifier

@TODO, circuit schematic of gain stage with parts highlighted In the RNN circuit model, we treat the Gain Stage as a black box with a single user-facing control (the "Gain" control). The RNN is designed to completely replace the Gain Stage in the circuit model. In the traditional circuit model, we use nodal analysis to model the amplifier stage, and summing amplifier circuits. For FF-2 and the clipping stage, we use a wave digital filter. Since FF-1 and the pre-amp circuit share a capacitor, we construct a joint WDF model of these two circuits, using the voltage output from the pre-amp circuit as the input to the amplifier stage, and the current output from FF-1 (summed with the current outputs of FF-2 and the clipping stage) as the input to the summing amplifier.

4.1. Audio Plugin

Digital audio effects are often implemented as audio plugins that can be used by mixing engineers, producers, and musicians in a consumer digital audio workstation (DAW) software. Common plugin formats include the Avid Audio Extension (AAX), Steinberg's Virtual Studio Technology (VST), and Apple's Audio Unit (AU) for desktop use, as well as Apple's Audio Unit v3 (AUv3) for mobile use. The JUCE C++ framework[1] is commonly used to create cross-platform, cross-format plugins.

[1]https://github.com/juce-framework/JUCE

Figure 6: *Audio plugin implementation of the Klon Centaur circuit model. Note controls for "Gain", "Treble", and "Level" analogous to the original circuit, as well as the "neural" parameter to control whether the emulation uses the traiditional circuit model, or the RNN model.*

As a demonstration of the two circuit emulations, we construct an audio plugin containing both models, allowing the user to switch between the two models for comparison. The plugin is implemented using JUCE/C++, along with a real-time Wave Digital Filter library[2] for the WDF models. While the current model uses custom implementations of the GRU and fully connected layers needed for the RNN model, in the future, we plan to upgrade this model to use the implementations provided by the Tensorflow Lite library[3].

4.2. Embedded Implementation

Digital audio effects are sometimes implemented on embedded devices for use in stage performances, often in the form of a guitar pedal, or synthesizer module. Deploying an audio effect on an embedded device can be difficult, due to the constraints in processing power and memory availability. Further, in order to achieve a more expressive performance, musicians often prefer effects that add minimal latency to the signal, meaning that the embedded implementation must be able to run with a very small buffer size.

We chose the Teensy 4.0 microcontroller as our embedded platform, since it contains a reasonbly powerful floating point processor, at a relatively low price point. The Teensy can be purchased along with an Audio Shield, which provides 16-bit stereo audio input/output at 44.1 kHz sampling rate. The Teensy has gained popularity in the audio community due to the Teensy Audio Library[4] that contains useful audio DSP functionality, as well as the Faust programming language which allows audio effects and synthesizers made in Faust to be exported for use on the Teensy [21].

[2]https://github.com/jatinchowdhury18/WaveDigitalFilters
[3]https://www.tensorflow.org/lite/
[4]https://www.pjrc.com/teensy/td_libs_Audio.html

5. RESULTS

asdjfahlsk

6. CONCLUSION

In this writing...

7. ACKNOWLEDGMENTS

The author would like to thank Pete Warden and the EE292 class at Stanford University for inspiring this project, as well as Julius Smith, Kurt Werner, and Jingjie Zhang for assistance with Wave Digital Filter modelling.

8. REFERENCES

[1] West Warren, "Builder Profile: Klon's Bill Finnegan," *Premier Guitar*.

[2] "Klon centaur analysis," *ElectroSmash*.

[3] Francois Germain, *Non-oversampled physical modeling for virtual analog simulations*, Ph.D. thesis, Stanford University, 2019.

[4] D.T. Yeh, *Digital Implementation of Musical Distortion Circuits by Analysis and Simulation*, Ph.D. thesis, Stanford Univeristy, 06 2009.

[5] Antoine Falaize and Thomas Helie, "Passive guaranteed simulation of analog audio circuits: A port-hamiltonian approach," *Applied Sciences*, vol. 6(10), pp. 273, 09 2016.

[6] A. Fettweis, "Wave digital filters: Theory and practice," *Proceedings of the IEEE*, vol. 74, no. 2, pp. 270–327, Feb 1986.

[7] Kurt James Werner, *Virtual Analog Modeling of Audio Circuitry Using Wave Digital Filters*, Ph.D. thesis, Stanford Univeristy, 06 2016.

[8] Martin Holters and Udo Zolzer, "A generalized method for the derivation of non-linear state-space models from circuit schematics," 09 2015.

[9] Julius O. Smith, *Spectral Audio Signal Processing*, `http://ccrma.stanford.edu/˜jos/sasp/`, accessed 2020-5-22, online book, 2011 edition.

[10] Eero-Pekka Damskagg, Lauri Juvela, and Vesa Valimaki, "Real-time modeling of audio distortion circuits with deep learning," in *Proc. of the 16th Sound and Music Computing Conference*, 2019.

[11] Julian D. Parker, Fabian Esqueda, and Andre Bergner, "Modelling of nonlinear state-space systems using a deep neural network," in *Proc. of the 22nd Int. Conference on Digital Audio Effects*, 2019.

[12] Alec Wright, Eero-Pekka Damskagg, and Vesa Valimaki, "Real-time black-box modelling with recurrent neural networks," in *Proc. of the 22nd Int. Conference on Digital Audio Effects*, 2019.

[13] Edward W. Maby, *Solid State Electronics*.

[14] Julius O. Smith, *Physical Audio Signal Processing*, `http://ccrma.stanford.edu/˜jos/pasp/`, accessed 2020-5-22, online book, 2010 edition.

[15] Chung-Wen Ho, A. Ruehli, and P. Brennan, "The modified nodal approach to network analysis," *IEEE Transactions on Circuits and Systems*, vol. 22, no. 6, pp. 504–509, 1975.

[16] Marco A. Martínez Ramirez and Joshua D. Reiss, "Modeling of nonlinear audio effects with end-to-end deep neural networks," *arXiv e-prints*, p. arXiv:1810.06603, Oct. 2018.

[17] Jatin Chowdhury, "Complex nonlinearities for audio signal processing," `https://ccrma.stanford.edu/~jatin/papers/Complex_NLs.pdf`, Feb 2020.

[18] François Chollet et al., "Keras," `https://github.com/fchollet/keras`, 2015.

[19] Kyunghyun Cho, Bart van Merrienboer, Çaglar Gülçehre, Fethi Bougares, Holger Schwenk, and Yoshua Bengio, "Learning phrase representations using RNN encoder-decoder for statistical machine translation," *CoRR*, vol. abs/1406.1078, 2014.

[20] Diederik P. Kingma and Jimmy Ba, "Adam: A method for stochastic optimization," *CoRR*, vol. abs/1412.6980, 2015.

[21] Romain Michon, Yann Orlarey, Stéphane Letz, and Dominique Fober, "Real time audio digital signal processing with faust and the teensy," in *Proc. of the 16th Sound and Music Computing Conference*, 2019.