DIGITIZING THE IBANEZ WEEPING DEMON WAH PEDAL

Chet Gnegy, Kurt James Werner

Center for Computer Research in Music and Acoustics (CCRMA), Stanford University
660 Lomita Drive, Stanford, CA 94305, USA
[chet|kwerner]@ccrma.stanford.edu

ABSTRACT

Being able to transform an analog audio circuit into a digital model is a big deal for musicians, producers, and circuit benders alike. In this paper, we address some of the issues that arise when attempting to make such a digital model. Using the canonical state variable filter as the main point of interest in our schematic, we will walk through the process of making a signal flow graph, obtaining a transfer function, and making a usable digital filter. Additionally, we will address an issue that is common throughout virtual analog literature; reducing the very large expressions for each of the filter coefficients. Using a novel factoring algorithm, we show that these expressions can be reduced from thousands of operations down to tens of operations.

1. INTRODUCTION

The *Weeping Demon* may be one of the most versatile wah pedals on the market. Much like the *Dunlop Crybaby*, the *Weeping Demon* offers control over the center frequency and Q of the filter. Due to the design of the *Weeping Demon*'s filter circuit, there is independent control over both of these features, as well as a controllable low range boost and a mode switch that changes the frequency range of the wah making it more suitable for the bass guitar. In this work, we model the filter circuit by obtaining its transfer function parameterized by its electrical components. The transfer function is then digitized via the bilinear transform [1][1].

The transfer function is obtained by reverse-engineering the circuit, which happens to be very similar to the canonical state variable filter (SVF) [2] consisting of four op-amp circuits in feedback with each other. The low impedance output of each op-amp allows us to treat each op-amp circuit independently and form a block diagram consisting of adds and multiplies. The transfer function at any node of the circuit can be obtained via Mason's rule. SVFs, which will be discussed in more detail later, have the interesting property that the high-, band-, and low-passed outputs are produced at each of three op-amps. In the *Weeping Demon*, a fourth op-amp combines the band- and low-pass outputs to produce a resonant low-pass filter, as is common for wah pedals.

Unfortunately, the transfer function that we obtain from Mason's rule has extremely complicated coefficients. In the case of the *Weeping Demon*, a product-of-sum coefficient can have as many as one hundred addends, each consisting of about five multiplications. We present a method of polynomial simplification capable of reducing the number of add/multiply operations for a single coefficient from several hundred down to 50 or fewer. Using common subexpression extraction, a typical compiler trick, we can reduce this even further.

[1] https://ccrma.stanford.edu/~jos/pasp/Bilinear_Transformation.html

Wah pedals have previously been studied in virtual analog. Models based on fitting biquads to measured filter responses can capture a wah pedal's basic global behavior[2], but they lack the detail of virtual analog physical models. Holters and Zölzer studied the *Dunlop Crybaby* in a nodal DK framework, handling efficient parameter update in the context of changing coefficients by exploiting a Woodbury identity—their technique is applicable to state-space systems where the number of variable parts is low compared to the total number of parts [3]. This is closely related to a technique used by Dempwolf *et al.*, who handle complicated coefficient updates by exploiting a certain minimized matrix formulation [4]. Falaize-Skrzek and Hélie studied the Crybaby pedal from a port-Hamiltonian perspective [5].

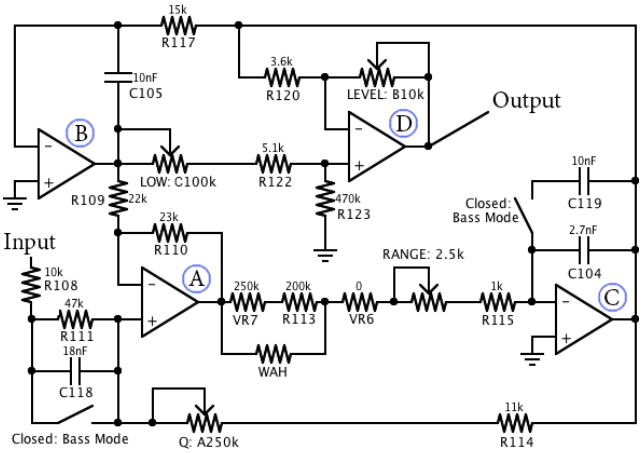

Figure 1: *The filter stage for the* Weeping Demon *with all components labeled.*

2. MODELING THE CIRCUIT

The filter stage of the *Weeping Demon* is shown in Figure 1. The switching circuit, which offers a bypass mode that turns on once the pedal has not been used for some programmed amount of time, will not be covered in this paper, mainly because there is no need for this in the digital model. The implementation of a timer that turns the effect off after a given time is trivial in software. Perhaps the most interesting bit about this circuit is the choice of sensors used to detect the angle of the foot pedal. Rather than using the common choice, a potentiometer, an optical sensor is used. An

[2] https://ccrma.stanford.edu/realsimple/faust_strings/Adding_Wah_Pedal.html

opaque fin is mounted to the bottom of the moving foot piece directly between an LED and a photoresistor. As the pedal is rocked, the fin allows more light to pass from the LED to the photoresistor, decreasing its resistance. The advantage of this is that unlike a potentiometer, the optical element will not accumulate dirt and become noisy with time and use. The modeling of the optical element will be discussed in section 3.

2.1. Overview

Prior to entering the shown circuit, the signal is passed through a buffer stage. It consists of two cascaded emitter followers, which provide a low impedance signal to the filter. The emitter follower stages act as a high-pass filter with cutoff of about 4 Hz. Because this is way below the audio range, will exclude it from the model.

The coupled network of op-amps shown is known as a state variable filter (SVF). It is rather complicated as a whole, but broken into components, it is much more easily understood. This decomposition will be used when deriving the transfer function. The first important observation is that op-amps B and C (labeled in the figure) are configured as integrators. The other two op-amps, A and D, are differential amplifiers. This is nearly enough information to dive into finding the transfer function, but it is insufficient for understanding how the circuit works.

We will start our analysis by noting that the important mechanics of the circuit can be realized by removing op-amp D from the circuit completely. This is because there is no feedback from D to any of the other stages. We can therefore ignore op-amp D, and remove the paths containing R_{120} and the potentiometer labeled LOW. This leaves us with a differential amplifier with non-inverting inputs from the filter input and from the output of C. Let us assume for now that the output of C is not feeding back to A. Equivalently, set the resistance of the potentiometer labeled Q to be arbitrarily high. We will relax this assumption later. This leaves with two cascaded integrators whose output is providing negative feedback to A. In essence, we are subtracting the original signal from itself.

The output of A and the output of B have a constant phase relationship: they are always 180 degrees out of phase. This should be easy to see, because the output of B appears at the inverting terminal of A. The integrators, B and C, are simply low-pass filters, each with a cutoff associated with some RC time constant. Without the feedback from B to A (basically ignoring the effects of A entirely), we can expect the output of B to have a low-pass characteristic. Now, considering the feedback, we note that we are adding an inverted, low-passed version of the input signal to itself. At low frequencies, we should expect a very minimal output of A. At high frequencies, the inverting terminal of A is very small in magnitude, so we see a high-pass behavior at the output of A. An important step in getting comfortable with this circuit may be to realize that by twice integrating a second order high-pass filter (A), we obtain a low-pass filter (C). As we reintroduce the feedback from the band-pass filter (B), we bring with it lower, more reasonable Q values [2].

2.2. Signal Flow Diagram

Using some basic topological information and superposition, we will represent the circuit as a block diagram with only adds and multiplies. First, lets simplify the component values, combining series and parallel elements into impedance terms, Z_i. We can see the reduced circuit in Figure 2. This will simplify the arithmetic quite a bit. The substitutions are shown in Table 1. It should be clear that we will derive a separate set of equations based on whether the mode switch is open or closed, designating Bass or Normal mode.

To figure out the global filtering properties of the SVF, it is useful to consider the op-amp stages individually. For each op-amp stage, we can derive how each of the other stages and the input voltage V_{in} contribute. These contributions, based on combined impedance values from Table 1 and the op-amp configurations, are summed together by superposition. This derivation, which is shown in detail for each stage in the next section, yields a global signal flow graph, as shown in Figure 3. We will show the derivation for the gains in the graph (the K_i's in Figure 3) in the next section.

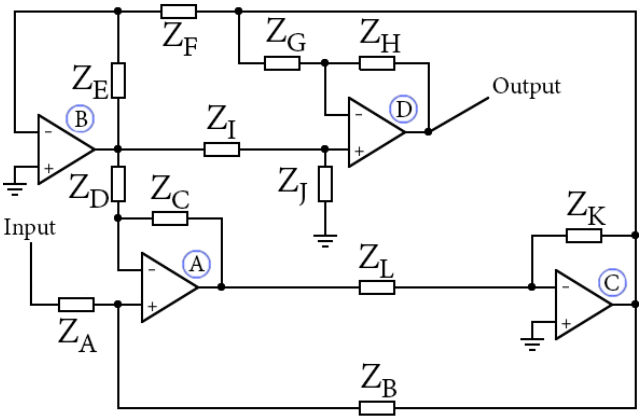

Figure 2: *The filter stage for the* Weeping Demon *with the components combined into impedance terms.*

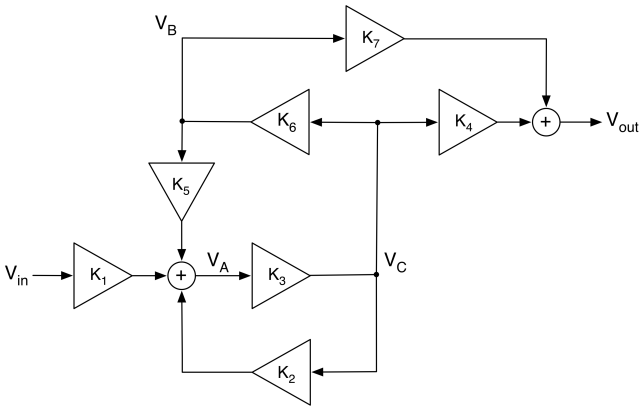

Figure 3: *A signal flow graph view of the SVF.*

2.3. Operational Amplifier Stages

Let's look at the four operational amplifier stages of the *Weeping Demon* Filter individually. These are the input summer, the first integrator, the second integrator, and the output summer.

Z_i	**Combined Components**	
	Bass	**Normal**
Z_A	R_{108}	$R_{108} + R_{111}\|\|\frac{1}{sC_{118}}$
Z_B	$R_Q + R_{114}$	
Z_C	R_{110}	
Z_D	R_{109}	
Z_E	$\frac{1}{sC_{105}}$	
Z_F	R_{117}	
Z_G	R_{120}	
Z_H	R_{LEVEL}	
Z_I	$R_{\text{LO}} + R_{112}$	
Z_J	R_{123}	
Z_K	$\frac{1}{s(C_{104}+C_{119})}$	$\frac{1}{s(C_{104})}$
Z_L	$(R_{\text{VR7}} + R_{113})\|\|R_{\text{WAH}} + R_{\text{VR6}} + R_{115} + R_{\text{RANGE}}$	

Table 1: *Substitutions for Z_i. Due to the switch in the circuit, we have different components for Z_A and Z_K depending on whether Bass or Normal mode is currently being used.*

Each stage can be analyzed by assuming ideal op-amps, invoking superposition, and using the equations for inverting amplifier and difference amplifiers. Ideal op-amps have zero output impedance—their outputs can be treated as ideal voltage sources. Furthermore, they are linear, so the output of each op-amp can be found by summing its response to each voltage source input. When finding the response to one voltage source input, the others are shorted. Under this condition, each op-amp acts as either a differential amplifier (with only non-inverting input) or an inverting amplifier to each input.

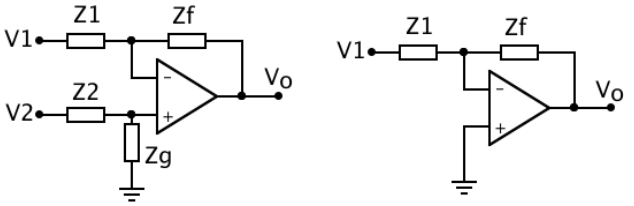

Figure 4: *Differential (left) and inverting (right) amplifiers with generalized impedances.*

Differential and inverting amplifiers with generalized impedances are shown in Figure 4. The output of each amplifier depends on the ratios between connected impedances. The output of a differential amplifier is given by:

$$V_o = -\frac{Z_f}{Z_1}V_1 + \left(\frac{Z_1 + Z_f}{Z_1}\right)\left(\frac{Z_g}{Z_g + Z_2}\right)V_2 \tag{1}$$

and the output of an inverting amplifier is given by:

$$V_o = -\frac{Z_f}{Z_1}V_1. \tag{2}$$

In all of our cases, when considering superposition, the inverting input V_1 of the differential amplifier will be grounded, so a simpler version of Eqn. (1) applies:

$$V_o = \left(\frac{Z_1 + Z_f}{Z_1}\right)\left(\frac{Z_g}{Z_g + Z_2}\right)V_2. \tag{3}$$

Table 2 shows the gains for the transfer function, op-amp, input voltage, configuration (**i**nverting or **d**ifferential), and impedances for each case of superposition.

op	TF	V_{in}	i/d	Z_1	Z_2	Z_f	Z_g
A	K_1	V_{in}	d	Z_D	Z_A	Z_C	Z_B
A	K_5	V_B	i	Z_D		Z_C	
A	K_2	V_C	d	Z_D	Z_B	Z_C	Z_A
B	K_6	V_C	i	Z_F		Z_E	
C	K_3	V_A	i	Z_L		Z_K	
D	K_7	V_B	d	Z_G	Z_I	Z_H	Z_J
D	K_4	V_C	i	Z_G		Z_H	

Table 2: *For each case of superposition, configuration and impedances seen by the op-amps.*

2.3.1. Input Summer

The input summer is the circuitry around op-amp A that combines V_{in}, the input to the SVF, with feedback from the first and second integrators (V_C and V_B). By superposition, we can get its output in terms of the partial transfer functions K_1, K_5, and K_2,

$$V_A = K_1 V_{\text{in}} + K_5 V_B + K_2 V_C, \tag{4}$$

where K_1, K_5, and K_2 are found in terms of the combined impedances using Eqns. (2)–(3) with values from Table 2 as appropriate:

$$K_1 = \left(\frac{Z_C + Z_D}{Z_C}\right)\left(\frac{Z_B}{Z_B + Z_A}\right) \tag{5}$$

$$K_5 = -\frac{Z_C}{Z_D} \tag{6}$$

$$K_2 = \left(\frac{Z_C + Z_D}{Z_D}\right)\left(\frac{Z_A}{Z_A + Z_B}\right). \tag{7}$$

In a standard SVF, this input stage would be purely resistive—the magnitude response would be flat and the resistances chosen to get the desired circuit response. In the *Weeping Demon*, there is a reactive component when the pedal is set in "normal mode." Since this is part of Z_A, it has implications for K_1 and K_2—how the summer affects the input signal V_{in} and also the feedback from the first integrator V_C. We can speculate about why this capacitor is put in place for "normal mode." Perhaps for a guitar input, the circuit designers wanted to damp down low frequencies for noise reasons; perhaps it is just an ad hoc voicing choice.

The output of the input summer (A) is the high-pass output of the SVF.

2.3.2. First Integrator

The first integrator is the circuitry around op-amp C that integrates the output V_A of the input summer. Its output V_C is applied to the second integrator, through a feedback path to the non-inverting input of the input summer, and to the inverting input of the output

summer. In terms of the circuit generalized impedances, its output is

$$V_C = K_6 V_A, \tag{8}$$

where

$$K_6 = -\frac{Z_K}{Z_L}. \tag{9}$$

Impedance Z_K changes depending on whether the pedal is set in "normal mode" or "bass mode." An extra capacitor is placed in parallel for "bass mode," shifting the response of the integrator. The integrator response is also shifted when the wah pedal's position changes, changing R_{WAH} and hence R_L; this is mechanism by which R_{WAH} shifts the output resonant filter's center frequency.

The output of the first integrator (V_C) is the band-pass output of the SVF.

2.3.3. Second Integrator

The second integrator is the circuitry around op-amp B that integrates the output V_C of the first integrator. Its output V_B is applied through a feedback path to the inverting input of the input summer and to the output summer. In terms of the circuit generalized impedances, its output is

$$V_B = K_3 V_C, \tag{10}$$

where

$$K_3 = -\frac{Z_E}{Z_F}. \tag{11}$$

The output of the second integrator (B) is the low-pass output of the SVF.

2.3.4. Output Summer

The output summer sums contributions from the band-pass (V_C) and low-pass (V_B) outputs of the SVF. By superposition, its output is:

$$V_{\text{out}} = K_7 V_B + K_3 V_C \tag{12}$$

where

$$K_7 = \left(\frac{Z_G + Z_H}{Z_G}\right)\left(\frac{Z_J}{Z_I + Z_J}\right) \tag{13}$$

$$K_3 = -\frac{Z_H}{Z_G}. \tag{14}$$

This combination of the band-pass (C) and low-pass(B) SVF outputs forms a resonant low-pass filter, a typical goal of wah pedal design. We note that it would be possible to modify the output summer circuitry (or the digital model) so that the output summer combined different SVF outputs, yielding other filter configurations like resonant high-pass, etc.

3. MODELING THE OPTICAL ELEMENT

The optical element that controls the center frequency of the wah resonance is the most difficult element in the circuit to model. Whereas resistors and capacitors in the circuit are well modeled in the audio band by their ideal generalized impedances and operational amplifiers can be considered ideal, the effect of the optical element is more complex.

The current–voltage (i–v) characteristics of the stationary LED and photoresistor pair depends in complex ways on the internal geometry of the pedal as well as the electrical characteristics of the LED and photoresistor. The optical/geometric properties of the enclosure are too complex to predict from first principles. Lacking datasheets or documentation for the LED and photoresistor, and even access to a good model of this photoresistor's behavior, the behavior of this pair of components is difficult to predict. So, we make recourse to black-box modeling and fit a model to measured data.

When the foot pedal is rocked, the black, metal fin that divides the LED and the photoresistor moves, and is no longer an obstruction. We cannot expect to get a useful measurement of the mapping from pedal angle to resistance with the pedal disassembled because of the ambient light for the room. We instead cut the copper traces around the photoresistor and solder wires to each of its terminals. The pedal is reassembled with the wires running out of the pedal through a small hole near the battery holder. Making no assumptions about the linearity of this element with respect to any of its parameters, we set out to measure its i–v transfer characteristic as a function of pedal angle, θ. The photoresistor is connected in series with a 33kΩ resistor via the long wires as seen in Figure 5. We sweep the voltage V_{DC} across the series connection, recording the voltage across the photoresistor ($V_{DC} - v_R$, where v_R is voltage across the 33kΩ resistance) as well as the current through the circuit, $v_R/33k\Omega$. θ is increased in increments of 2°, as measured from the rotational axis. For each θ, we sweep V_{DC} from 0V up to around 5V. Figure 6 shows the result.

For each angle θ, we observe an approximately straight line in the i–v characteristic, indicating that the photoresistor can be accurately modeled as linear for any given θ value. The slope of this line gives us the resistance of the component. The relationship between θ and the resistance of the photoresistor is shown in Figure 7. We can now model the resistance $R_{\text{WAH}}(\theta)$ as the pedal is rocked by fitting a curve to these points.

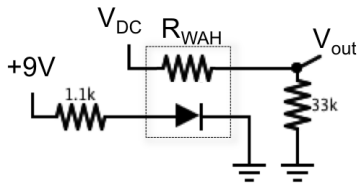

Figure 5: *Setup to find $R_{WAH}(\theta)$.*

4. THE TRANSFER FUNCTION

Now that we have a signal flow chart, we can get our transfer function. It is of interest to express the transfer function as a ratio of two polynomials in s, as seen in Equation 15.

$$H(s) = \frac{b_m s^m + \cdots + b_2 s^2 + b_1 s + b_0}{a_n s^n + \cdots + a_2 s^2 + a_1 s + a_0} \tag{15}$$

Both Matlab and Python symbolic libraries are used to obtain and reduce the transfer function. We do this via Mason's gain law [6, 7], the same treatment seen in modeling the TR-808 cowbell [8] and as Kramer demonstrates generically [9]. In short, we leverage a Matlab function written by Rob Walton[3] that converts

[3] `http://www.mathworks.com/matlabcentral/fileexchange/22-mason-m`

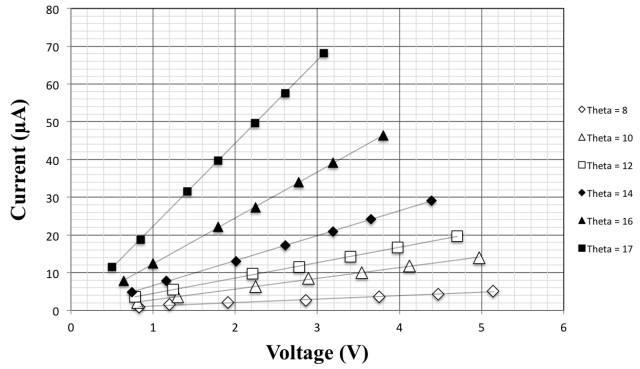

Figure 6: *Current–voltage characteristic for different pedal positions. $\theta = 6°$ and $\theta = 8°$ produced the same curve indicating that the internal geometry didn't change much. The maximum angle without putting a lot of pressure on the foot pedal was $\theta = 17°$.*

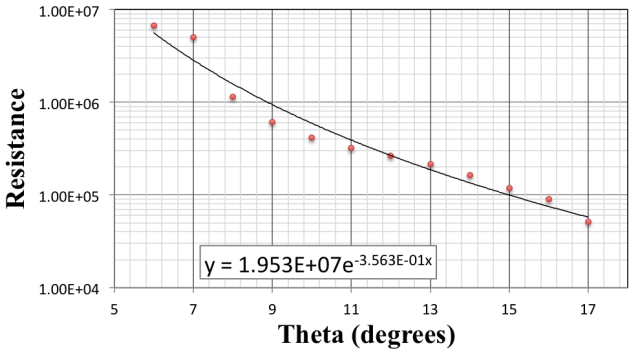

Figure 7: *The curve produced for $R_{WAH}(\theta)$. A curve fitted equation is shown. Different curve fits would change the behavior of the model significantly. The presented equation was used for simplicity and because it matched the real pedal decently well. After some experimentation, a higher order fit was not deemed necessary.*

a netlist for a network of summations and gains to a single transfer function. This can be done for any chosen output of the filter, but we perform our analysis for V_{out}. The transfer function seen in Equation 16 expresses the transfer characteristic from input to output in terms of the gain coefficients, K_i. The impedance values, Z_i in Table 2, can now be substituted back for the $K's$, and in turn, the symbols designating the component values.

$$H(s) = \frac{V_{\text{out}}}{V_{\text{in}}} = \frac{-K_1K_3(K_4 + K_6K_7)}{K_2K_3 + K_3K_5K_6 - 1} \tag{16}$$

We do this using the `simplifyFraction` function in the Matlab symbolic library. The `simplifyFraction` function reduces a fraction such that the greatest common divisor of the numerator and denominator is 1. Without this step, the equation is quite messy and does not necessarily contain only positive powers of s. Once we have done that, we are left with an analog filter with coefficients that can be expressed in the form shown in Equation 15. In this form, it is clear that the normal mode filter is third order, and the bass mode filter is only second order.

The result is something familiar from other works in virtual analog: the coefficients are very complicated. Two examples of this are [8] and [10], in which Werner *et al.* demonstrate a discretization of a band-pass filter from the *TR-808* cowbell and Yeh and Smith discretizes the '59 *Fender Bassman Tone Stack*. Each of these works models a circuit with a relatively low component count and results in coefficients that require, in their presented form, 82 and 280 operations. The *Weeping Demon* in normal mode features a whopping 3917 operations in expanded form. Even the partially factored result given by Matlab contains nearly 400 operations. The worst of these coefficients, a_2, in the form given by Matlab, is shown:

$$\begin{aligned}
a_2 = {} & R_{120}(R_{\text{LO}} + R_{122} + R_{123})C_{105}R_{117}(\\
& C_{104}R_{\text{Q}}R_{109}R_{113}R_{115} + C_{104}R_{\text{Q}}R_{109}R_{113}R_{\text{RANGE}} + \\
& C_{104}R_{108}R_{109}R_{113}R_{115} + C_{104}R_{109}R_{111}R_{113}R_{115} + \\
& C_{104}R_{109}R_{113}R_{114}R_{115} + C_{104}R_{109}R_{113}R_{114}R_{\text{RANGE}} + \\
& C_{118}R_{108}R_{110}R_{111}R_{113} + C_{104}R_{108}R_{109}R_{113}R_{\text{RANGE}} + \\
& C_{104}R_{109}R_{111}R_{113}R_{\text{RANGE}} + C_{118}R_{108}R_{109}R_{111}R_{113} + \\
& C_{104}R_{\text{Q}}R_{109}R_{113}R_{\text{VR6}} + C_{104}R_{109}R_{111}R_{\text{RANGE}}R_{\text{VR7}} + \\
& C_{104}R_{\text{Q}}R_{109}R_{\text{RANGE}}R_{\text{VR7}} + C_{104}R_{\text{Q}}R_{109}R_{113}R_{\text{WAH}} + \\
& C_{104}R_{\text{Q}}R_{109}R_{115}R_{\text{WAH}} + C_{104}R_{108}R_{109}R_{113}R_{\text{VR6}} + \\
& C_{104}R_{108}R_{109}R_{115}R_{\text{VR7}} + C_{104}R_{109}R_{111}R_{113}R_{\text{VR6}} + \\
& C_{104}R_{109}R_{111}R_{115}R_{\text{VR7}} + C_{104}R_{109}R_{113}R_{114}R_{\text{VR6}} + \\
& C_{104}R_{109}R_{114}R_{115}R_{\text{VR7}} + C_{118}R_{108}R_{109}R_{111}R_{\text{VR7}} + \\
& C_{118}R_{108}R_{110}R_{111}R_{\text{VR7}} + C_{104}R_{\text{Q}}R_{109}R_{\text{RANGE}}R_{\text{WAH}} + \\
& C_{104}R_{108}R_{109}R_{\text{RANGE}}R_{\text{VR7}} + C_{104}R_{\text{Q}}R_{109}R_{115}R_{\text{VR7}} + \\
& C_{104}R_{109}R_{114}R_{\text{RANGE}}R_{\text{VR7}} + C_{104}R_{108}R_{109}R_{113}R_{\text{WAH}} + \\
& C_{104}R_{108}R_{109}R_{115}R_{\text{WAH}} + C_{104}R_{109}R_{111}R_{113}R_{\text{WAH}} + \\
& C_{104}R_{109}R_{111}R_{115}R_{\text{WAH}} + C_{104}R_{109}R_{113}R_{114}R_{\text{WAH}} + \\
& C_{104}R_{109}R_{114}R_{115}R_{\text{WAH}} + C_{118}R_{108}R_{109}R_{111}R_{\text{WAH}} + \\
& C_{118}R_{108}R_{110}R_{111}R_{\text{WAH}} + C_{104}R_{108}R_{109}R_{\text{RANGE}}R_{\text{WAH}} + \\
& C_{104}R_{109}R_{111}R_{\text{RANGE}}R_{\text{WAH}} + C_{104}R_{\text{Q}}R_{109}R_{\text{VR6}}R_{\text{WAH}} + \\
& C_{104}R_{\text{Q}}R_{109}R_{\text{VR6}}R_{\text{VR7}} + C_{104}R_{109}R_{114}R_{\text{RANGE}}R_{\text{WAH}} + \\
& C_{104}R_{\text{Q}}R_{109}R_{\text{VR7}}R_{\text{WAH}} + C_{104}R_{108}R_{109}R_{\text{VR6}}R_{\text{VR7}} + \\
& C_{104}R_{109}R_{111}R_{\text{VR6}}R_{\text{VR7}} + C_{104}R_{109}R_{114}R_{\text{VR6}}R_{\text{VR7}} + \\
& C_{104}R_{108}R_{109}R_{\text{VR6}}R_{\text{WAH}} + C_{104}R_{108}R_{109}R_{\text{VR7}}R_{\text{WAH}} + \\
& C_{104}R_{109}R_{111}R_{\text{VR6}}R_{\text{WAH}} + C_{104}R_{109}R_{111}R_{\text{VR7}}R_{\text{WAH}} + \\
& C_{104}R_{109}R_{114}R_{\text{VR6}}R_{\text{WAH}} + C_{104}R_{109}R_{114}R_{\text{VR7}}R_{\text{WAH}} \\
&)
\end{aligned}$$

5. COMPARISON TO SPICE AND REAL DATA

When we compare the (analog) frequency response to the SPICE model, they are a perfect match. The digital frequency response, which is obtained via a third order bilinear transform [11], is shown in Figure 8 alongside the SPICE model. As expected, we see a sharp roll-off associated with the frequency warping characteristic of the bilinear transform near the Nyquist limit. This gives us assurance that we are modeling the schematic effectively. Indeed, we see that when we compare using any set of parameters, we have a model that is consistent with SPICE.

Though a full discussion is outside the scope of this paper, the match between the model and measurements of the real pedal is not as good. In general, the match is only approximate, showing discrepancies in center frequency, Q, and overall gain of the transfer function. This error can potentially be ascribed to limitations in our model of the optical element, which shows sensitivity to curve-fitting of the $R_{\text{WAH}}(\theta)$ relationship. 5% changes in curve

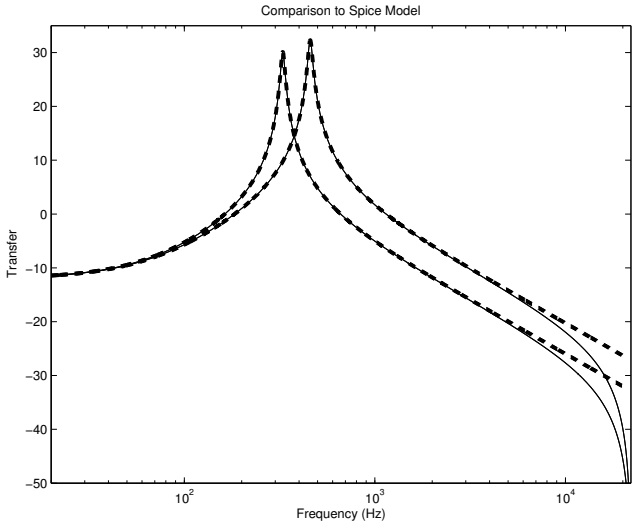

Figure 8: *Comparison to the SPICE Model. The curve for our model is the solid line, and the SPICE model is shown as a dashed line. The curves are shown for a pair of different parameter values. The curves are no longer overlapping in the high frequencies because of the frequency warping due to the bilinear transform.*

parameters resulted in changes of R_{WAH} on the order of tens of megaohms. Board-tracing errors are also possible, but unlikely since both authors independently traced the same schematic, and the schematic shows such a close match to a standard and sensible design (the SVF). Component, manufacturing, and trimpot tuning tolerances seem to be a likely source of error. Comparisons between 3 different *Weeping Demon* pedals showed that each pedal had a very different voicing, and that 2 of the 3 even exhibited instability when the Q knob was turned past about 2 o'clock. This was discovered within minutes of taking the pedals out of the box and is strongly in support of either a wide degree of variation between pedals or just a poor filter design.

Though we lack a strong case that the digital model matches the real-world circuit with a high degree of accuracy, the digitization scheme remains valid for the traced schematic, as shown by the correspondence with SPICE. In fact, a main finding of this paper is not the digital model itself, but a more general technique for minimized computation effort of digital filter coefficients which will be presented in the next section.

6. COEFFICIENT REDUCTION

The reduction process is done in two steps and was implemented using the symbolic Python library, SymPy [12]. First, there is a factoring step, and second is the common subexpression extraction step. To further motivate the need for the factoring step, it is important to mention that neither Matlab nor Python's symbolic packages would provide adequate simplification to the some of the more difficult factoring problems. If there was, say, a resistance value that was included in every term of a long coefficient computation, however, it would successfully factor it out. As a result, the coefficients returned by Matlab are not completely expanded and do have minimal amounts of factoring. The coefficient above, a_2, is an example of this. Our algorithm provides a much more satisfactory result, though no claims of optimality will be made.

The factoring step is a recursive algorithm that is shown in pseudocode in Algorithm 1. Before getting into the details of the main algorithm, it is necessary to introduce some supplementary functions (most of which are built into SymPy).

```
// Expands any parenthetical groupings;
function expand(expression)

// Finds all symbols that are included in 'expression';
function getVariables(expression)

// Finds the order of 'expression' with respect to 'var';
function orderOf(expression, var)

// Collects the coefficients of 'expression' for all power;
// of the variable 'var';
function collectTerms(expression, var)

// Picks a variable from a list 'vars' given some heuristic;
function chooseVar(vars)

// The main factoring algorithm;
function factored(exp)
    terms = [];
    // get all variables contained in exp;
    vars = getVariables (exp);
    if length(vars) ≤ 1 then
        terms.append( (exp,1) );
    else
        pickVar = chooseVar (vars);
        N = orderOf (exp, pickVar);
        // pows: array of descending powers of pickVar;
        // coeffs: coefficients to terms in pows;
        (coeffs, pows) = collectTerms (exp, pickVar);
        for i=0 to N do
            coeffs[i] = factored (coeffs[i]);
            terms.append( (pows[i], coeffs[i]) );
        end
    end
    recombine = 0;
    for i=0 to length(terms) do
        recombine += terms[i][0]*terms[i][1]
    end
    return recombine;
```

Algorithm 1: *The factoring algorithm used to reduce the expressions to a more computationally efficient form.*

The first, `expand`, is used to remove all parenthetical expressions by expansion. For example, $f(a,b,c) = (a+b)(c+b)$ is expanded to get $ac + ab + bc + b^2$. If we would like to find the variables involved in $f(a,b,c)$, we can access them via the `getVariables` function. `getVariables` will, in this case, return the list $[a,b,c]$.

The `orderOf` function returns the highest power of an expression with respect to a single variable.

The `collectTerms` function is used to factor out a single variable. For example, `collectTerms`$(f(a,b,c),b)$ will return two lists; the first of which containing the coefficients for each power of b contained in $f(a,b,c)$, and the second containing those powers of b. For $f(a,b,c)$ and (b), the returned lists are $[1, a+c, ac]$ and $[b^2, b, 1]$. The inner product of these two lists will, by definition, give an expression that is mathematically equivalent to $f(a,b,c)$. However, without additional expansion,

they are not identically equivalent and the inner product of the result is guaranteed to have *at most* the same number of operations as $f(a, b, c)$.

Algorithm 1 shows the process by which we reduce the equations. The basic premise of the algorithm is that we pick a variable v_i from the expression using some heuristic and factor it out. This choice is accomplished using the `chooseVar` function, whose implementation will be discussed shortly. We then factor by performing `collectTerms` on the expression. For each coefficient of v_i's powers (including $v_i^0 = 1$), we recurse. A list of the terms that are being factored is kept. Once each coefficient to v_i's powers is factored, we will recombine them using an inner product as previously mentioned, noting that this form is guaranteed to have, in the worst case, just as many operations as it started with. Once the expression is a function of only a single variable, nothing more can be factored out and the recursion ceases.

For a given expression that is completely expanded, we are guaranteed to get a result that is at least as good as the original. However, we note that the choice of v_i will change the amount of possible simplification. Thus, a good heuristic for choosing v_i is needed to ensure that partially factored input will still be improved by this algorithm. Three heuristics were tested for the `chooseVar` implementation: choosing the first variable found, choosing the most common variable, and choosing the least common variable. The most common variables is taken to mean "most common from the original set of expressions" and not from the current subexpression (the symbol, exp, from Algorithm 1). In the event that the most common variable is not in the subset, $vars$, it chooses the most common variable from the expression that does appear in $vars$. The same method was used for the least common variable heuristic. Choosing the first variable is subject to whatever ordering that Matlab may put on the variables, but we will assume that this heuristic is making a fairly arbitrary choice. Of the three, choosing the most common variable performed the best on each of the tested sets of coefficients and will be used when reporting results, followed closely by the arbitrary choosing of the first variable.

Once the factoring algorithm has completed, the common subexpression extraction (CSE) step is performed. This step relies completely on the `cse` function built into SymPy. This function takes an expression or group of expressions and replaces any calculations that happen multiple times with a temporary variable whose value is computed only once. This creates several more expressions than we started with, but the total number of operations will be reduced with every substitution. The algorithm then stores a copy of the expressions and counts the operations. The results at each step are shown in Table 3. The calculations for the coefficients are shown in Equations 17 and 18. The initial state of the coefficients (partially factored or not) was not observed to change the final operation counts. By inspection, we see that the *Weeping Demon* could be simplified slightly further, saving a few operations. For instance, the b_2 coefficient contains the term $x_2x_3x_5$, which should have been replaced with x_6. This is clearly a fault of the CSE algorithm, but it does not change the fact that taking the approach of factoring and using CSE, even with off-the-shelf software, leads to dramatic reductions in the required computation.

It is interesting to note that we see much larger reductions from the fully expanded form of the *Weeping Demon* equations than for the tone stack and cowbell. It is speculated that the reason is due to the isolation between op-amps in the circuit topology. The tone stack was an impedance network with no "stages" to be treated separately. Similarly, the band-pass filter of the cowbell was only a single op-amp. There was less to condense and no isolated stages to prevent terms from combining in complicated ways.

Operations Count				
	Initial	Expanded	Factored	CSE
WD: Normal Mode	383	3917	140	66
WD: Bass Mode	75	2410	64	40
Tone Stack	280	312	160	86
808 Cowbell	82	82	57	32

Table 3: *The number of operations for the analog coefficients of several virtual analog models. Initial counts for the Weeping Demon use the forms given by Matlab and initial counts for the Fender Bassman tone stack and TR-808 cowbell are for the form of the equations presented in each authors' original work. Expanded operation counts are obtained by completely expanding each coefficient. The operation count after the factoring algorithm shows modest improvement and finally after the CSE step the operation count is much lower.*

$$
\begin{aligned}
x_0 &= R_{\text{LO}} + R_{122} + R_{123} \\
x_1 &= C_{105}R_{111}R_{117}x_0 \\
x_2 &= R_{\text{Q}} + R_{114} \\
x_3 &= R_{109} + R_{110} \\
x_4 &= R_{113} + R_{\text{VR7}} \\
x_5 &= R_{\text{WAH}} + x_4 \\
x_6 &= x_2x_3x_5 \\
x_7 &= C_{118}R_{111}R_{123} \\
x_8 &= C_{105}R_{117} \\
x_9 &= x_0x_8 \\
x_{10} &= R_{108} + x_2 \\
x_{11} &= R_{115} + R_{\text{RANGE}} + R_{\text{VR6}} \\
x_{12} &= R_{\text{WAH}}(x_{11} + x_4) + x_{11}x_4 \\
x_{13} &= R_{110}R_{111} \\
x_{14} &= C_{118}R_{108}x_5 \\
x_{15} &= C_{104}x_{12} \\
x_{16} &= R_{120}x_0x_5 \\
x_{17} &= R_{108} + R_{111} \\
\boldsymbol{b_2} &= C_{118}R_{\text{LEVEL}}x_1x_2x_3x_5 \\
\boldsymbol{b_1} &= x_6(R_{\text{LEVEL}}(x_7 + x_9) + R_{120}x_7) \\
\boldsymbol{b_0} &= R_{123}x_6(R_{\text{LEVEL}} + R_{120}) \\
\boldsymbol{a_3} &= C_{104}C_{118}R_{109}R_{120}x_1x_{10}x_{12} \\
\boldsymbol{a_2} &= R_{120}x_9(R_{109}(R_{111}(x_{14} + x_{15}) + x_{10}x_{15}) + x_{13}x_{14}) \\
\boldsymbol{a_1} &= x_{16}(C_{118}x_{10}x_{13} + x_{17}x_3x_8) \\
\boldsymbol{a_0} &= R_{110}x_{16}(x_{17} + x_2)
\end{aligned}
\tag{17}
$$

$$
\begin{aligned}
x_0 &= R_{\text{LO}} + R_{122} + R_{123} \\
x_1 &= C_{105}R_{117}x_0 \\
x_2 &= R_{\text{Q}} + R_{114} \\
x_3 &= R_{109} + R_{110} \\
x_4 &= R_{113} + R_{\text{VR7}} \\
x_5 &= R_{\text{WAH}} + x_4 \\
x_6 &= R_{108} + x_2 \\
x_7 &= R_{115} + R_{\text{RANGE}} + R_{\text{VR6}} \\
\boldsymbol{b_1} &= R_{\text{LEVEL}}x_1x_2x_3x_5 \\
\boldsymbol{b_0} &= R_{123}x_2x_3x_5(R_{\text{LEVEL}} + R_{120}) \\
\boldsymbol{a_2} &= R_{109}R_{120}x_1x_6(C_{104} + C_{119})(R_{\text{WAH}}(x_4 + x_7) + x_4x_7) \\
\boldsymbol{a_1} &= C_{105}R_{108}R_{117}R_{120}x_0x_3x_5 \\
\boldsymbol{a_0} &= R_{110}R_{120}x_0x_5x_6
\end{aligned}
\tag{18}
$$

7. CONCLUSIONS

We have presented a method for going from a schematic of a state variable filter to a numerical model that well replicates the behavior of the SPICE model. Though much of the analysis was specific to the state variable filter, these methods could easily be adapted to other linear circuits. Additionally, we tackled the common problem of having very large expressions for the filter coefficients by doing a factoring algorithm followed by common subexpression extraction. The results of this factoring were quite promising as they can reduce the operation count for the filter coefficients from thousands down to tens. It is now much more computationally efficient to repeatedly compute coefficients for component values that may change as a function of time.

8. SOURCE CODE

Full schematics, SPICE models, source code, and illustrated documentation can be found on the web:
`http://www.chetgnegy.com/projects/weepingdemon.html`.

9. ACKNOWLEDGMENTS

Thanks to Jonathan Abel for advice, for sharing his extensive experience in virtual analog, and for just being a fun guy to talk to.

10. REFERENCES

[1] Julius O. Smith, *Physical Audio Signal Processing for Virtual Musical Instruments and Audio Effects*, 2010, online book, 2010 edition.

[2] Lawrence P Huelsman, Ed., *Active filters: lumped, distributed, integrated, digital, and parametric*, vol. 11 of *Inter-University Electronics Series*, McGraw–Hill, 1970.

[3] Martin Holters and Udo Zölzer, "Physical modelling of a wah-wah effect pedal as a case study for application of the nodal DK method to circuits with variable parts," in *Proceedings of the International Conference on Digital Audio Effects (DAFx-11)*, Paris, France, September 19–23 2011, pp. 31–35.

[4] Kristjan Dempwolf, Martin Holters, and Udo Zölzer, "Discretization of parametric analog circuits for real-time simulations," in *Proceedings of the International Conference on Digital Audio Effects (DAFx-10)*, Graz, Austria, September 6–10 2010, vol. 13.

[5] Antoine Falaize-Skrzek and Thomas Hélie, "Simulation of an analog circuit of a wah pedal: a port-Hamiltonian approach," in *Proceedings of the International Audio Engineering Society Convention*, New York, NY, October 17–20 2013, vol. 135.

[6] Samuel J. Mason, "Feedback theory—further properties of signal flow graphs," in *Proceedings of the IRE*, 1956.

[7] Samuel J. Mason, "Topological analysis of linear nonreciprocal networks," *Proceedings of the IRE*, vol. 45, no. 6, pp. 829–838, 1957.

[8] Kurt James Werner, Jonathan S. Abel, and Julius O. Smith, "More cowbell: a physically-informed, circuit-bendable, digital model of the TR-808 cowbell," in *Proceedings of the International Audio Engineering Society Convention*, Los Angeles, CA, October 9–12 2014, vol. 137.

[9] Martin Krämer, "Design of a CMOS sample-and-hold amplifier for a high precision front-end circuit using and extended $^{g_m}/_{I_{ds}}$ method," M.S. thesis, Technische Universität, Kaiserlautern, 2009.

[10] David Te-Mao Yeh and Julius O. Smith, "Discretization of the '59 Fender Bassman tone stack," in *Proceedings of the International Conference on Digital Audio Effects (DAFx-06)*, Montréal, Canada, September 18–20 2006.

[11] Peter J. Pupalaikis, "Bilinear transformation made easy," ICSPAT, 2000.

[12] "Sympy," `http://www.sympy.org/en/index.html`, Accessed: 2015-02-6.