AN EVOLUTIONARY ALGORITHM FOR AUTOMATIC GUITAR AMPLIFIER MODELING

Benjamin QUIÉDEVILLE[1], **Tanguy RISSET**[2], **Romain MICHON**[3], and **Stéphane LETZ**[1]

[1]**Grame**, Grame-CNCM, INSA Lyon, Inria, CITI, UR3720, 69621 Villeurbanne, France
[2]**INSA Lyon**, INSA Lyon, Inria, CITI, UR3720, 69621 Villeurbanne, France
[3]**INRIA**, Inria, INSA Lyon, CITI, UR3720, 69621 Villeurbanne, France

ABSTRACT

This paper introduces a grey-box approach for automatic guitar amplifier modeling based on evolutionary optimization. We propose a generic digital amplifier model implemented in the Faust programming language, combining domain knowledge of tube amplification with parameterized signal processing blocks. The model includes four nonlinear gain stages, tone shaping, and filtering components, resulting in a 33-parameter system. Parameter estimation is performed using an evolutionary strategy, guided by a compound loss function designed to capture spectral, temporal, and dynamic characteristics of the target amplifier. Experiments conducted on a dataset of 94 reference models demonstrate rapid convergence the proposed approach offers a computationally efficient alternative to neural methods and eliminates the need for manual parameter tuning.

1. INTRODUCTION

Guitar amplifier distortion is, to this day, still closely dependent on vacuum tube technology, even though it was made obsolete in the 1960s with the rise of transistors and integrated circuits. However, vacuum tubes are still used in guitar amplifiers because they are often operated outside their linear amplification range, where they exhibit saturation characteristics that differ significantly from those of transistors.

Vacuum tubes have numerous drawbacks compared to discrete transistors: they are bulky, consume more energy, and are less efficient (power amplifier tubes can dissipate significant heat). They also wear out quickly (typically one to two years for power tubes) and are expensive due to their niche status in the electronic components market. Developing alternatives to vacuum tubes enables more accessible devices and opens up new possibilities for designing innovative music-making tools.

For many years, numerous studies have attempted to model vacuum tube–based guitar amplifiers. With the advent of artificial intelligence, new tools have emerged that can *learn* how to model a given amplifier using neural networks. Neural Amp Modeler [1] is a well-known open-source example of such a tool. However, neural networks are known to be computationally intensive, and there is a growing trend toward using alternative optimization algorithms that may be more efficient for domain-specific problems, particularly through a grey-box approach. For instance, the use of a high-level, audio-specific language combined with genetic algorithms has not yet been explored.

This study presents a digital implementation of a generic grey-box guitar amplifier using the Faust programming language [2] and the Julia programming environment, aimed at emulating the sounds of a wide variety of guitar amplifiers. The parameters of the grey-box model are optimized using evolutionary algorithms so that they match a given hardware unit. The results are obtained with rapid convergence and demonstrate acceptable acoustic quality based on informal auditory evaluation.

The paper is organized as follows. First, we review prior work on guitar amplifier modeling, musical hardware emulation, and automatic optimization in music technologies. Then, we present the amplifier design, its implementation in the Faust programming language, and the genetic optimization of its parameters in detail. Finally, we present and discuss preliminary performance results.

2. PREVIOUS WORK

Different virtual analog modeling (VA) methods have been explored to digitally recreate the behavior of vacuum tubes, and several review studies have surveyed these approaches [3, 4]. White-box models aim to digitize the electrical equations of components in order to simulate them as closely as possible to their operating conditions. Extensive documentation has been produced by tube manufacturers for electrical engineers, providing the necessary equations to describe their behavior. These models have been used in various studies and integrated into different numerical schemes (e.g., Helie–Cohen models, [5–11]). Wave Digital Filters (WDF) [12, 13] and Port Hamiltonian Systems [14–16] both different approaches to modeling components and their nonlinear interactions.

In contrast, black-box models make no assumptions about the target system and instead rely on measured input–output signal transformations to optimize their parameters. These models can take the form of deep neural networks [1], as part of a broader body of work on deep learn-

Copyright: © 2026. This is an open-access article distributed under the terms of the Creative Commons Attribution 3.0 Unported License, which permits unrestricted use, distribution, and reproduction in any medium, provided the original author and source are credited.

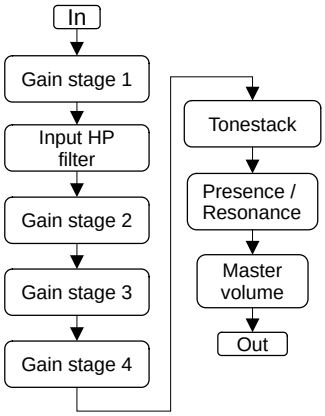


Figure 1. Full flow diagram of the amplifier model

ing for audio signal processing [17–19], or, in some cases, Volterra series formulations [20], which extend convolution kernels to nonlinear systems. Some industrial solutions [21–23] have resulted in products with extensive use in the studio as well as by touring musicians.

Grey-box models lie between these two approaches: they leverage partial knowledge of the target system to simplify the implementation while still relying on automatic optimization for parameter estimation. Such methods have been used in early industrial (e.g., patented approaches [24, 25]) and academic [26, 27] efforts to develop digital alternatives to vacuum tube-based guitar amplifiers.

Among machine learning and empirical methods, evolutionary algorithms are sometimes used to optimize and tune audio processing and synthesis models. Evolutionary algorithms [28] explore the parameter space more thoroughly than gradient-based ones by always generating new random solutions and selecting well-performing ones. This helps avoid local minima in the cost function as a new random solution can outperform previously selected ones. The subspace of evolutionary computing used in this study is detailed in Section 4.2. In audio applications, evolutionary algorithms have been used for complex system identification, such as musical instruments [29] and the human vocal tract [30], as well as for learning filter coefficients [31], sound matching [29, 32, 33], and natural reverberation design [34–36].

3. GUITAR AMPLIFIER MODEL

In this section, we present the design of a high-gain guitar amplifier model, with a particular focus on vacuum tube simulation. This four-stage gain model, illustrated in Fig. 1, is inspired by an audio plugin [37] that we wrote, which has proven to be flexible and effective. The four stages are implemented using the Faust programming language, which is well suited for audio algorithm exploration.

3.1 The Vacuum Tube

The triode vacuum tube amplifying stage is the main component of interest in this model, as it is responsible for generating most of the distortion and accounts for the majority of the computational complexity. A full schematic of a triode stage is shown in Fig. 2. The three electrodes of a preamplifier triode are the cathode, the plate, and the control grid. The cathode is connected to ground, and a filament heats it so that it emits electrons via thermionic emission. These electrons travel through the tube toward the plate, held at a high potential set by the supply voltage (typically between 200 V and 300 V), generating a large current. Before reaching the plate, the electrons pass through the control grid. A small negative voltage between the grid and the cathode (typically ranging from 0 V to -3 V) repels the electrons and reduces the output current. Thus, a small variation in the input voltage produces a large change in the output current, resulting in amplification.

When the grid voltage becomes more negative, it increasingly repels the electrons, reducing the current and therefore the voltage drop at the anode, which approaches the supply voltage. Conversely, when the grid voltage approaches 0 V, the electron flow is less impeded, leading to a larger voltage drop at the anode. As a result, the anode voltage has an inverted polarity with respect to the grid voltage.

Tube distortion occurs when the control grid voltage moves outside its normal operating range. Below -3 V, the grid almost completely repels the electron flow; no current passes, and the tube effectively ceases to operate (cutoff). Above 0 V, the cathode-to-grid voltage becomes positive, and a current flows into the grid. This current shifts the bias voltage toward more negative values, altering the tube's behavior and, at higher levels, producing a diode-like clipping effect (driving the signal to ground beyond a certain threshold). The bias voltage, set by the designer through the value of the cathode resistor, determines the operating point of the input signal and controls its sensitivity to voltage variations. A typical bias voltage is around -1.5 V (midway between 0 V and -3 V, the two distortion thresholds), although it may be set closer to -3 V to alter the saturation.

Additional elements impact the sound of the triode gain stage. Adding a capacitor across the cathode resistor reduces negative feedback in a frequency dependent fashion, reducing gain for low frequencies, thereby modifying the amplifier's response. Adding another capacitor across the plate resistor, forming a low-pass filter, is commonly used by designers to control high frequencies at each stage's output. The coupling capacitor, while primarily used to remove the DC offset induced by the plate supply voltage, also attenuates low frequencies. Finally, a voltage divider is added to each stage to attenuate the signal going into the next one; this is an important component to adjust, as it directly determines the amount of saturation in the amplifier.

3.2 Faust Implementation

This generic amplifier model is implemented in the *Faust* programming language [2]. This choice stems from its spe-

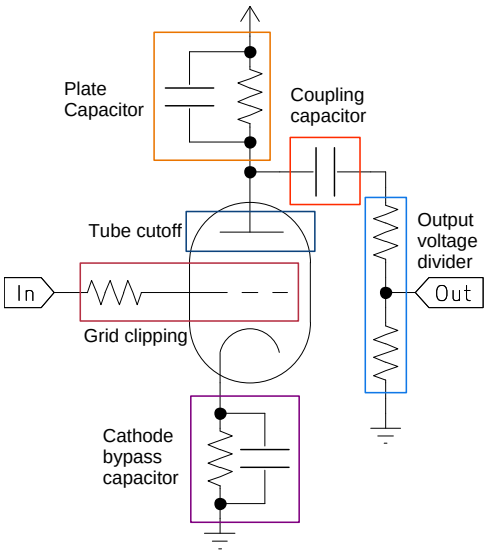


Figure 2. Triode Vacuum tube pre-amplifying stage

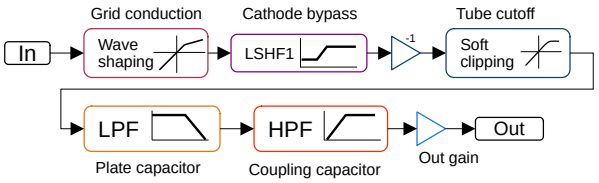


Figure 3. Flow diagram of the tube triode stage. *LSHF*: Low SHelf Filter, *LPF/HPF*: Low/High Pass Filter

cialization in audio programming and its ability to generate code in multiple formats. The same code can be exported to C, C++, WebAssembly, LLVM IR or Rust for use in real-time applications and, in this study, to Julia [38] for training using its rich ecosystem of scientific computing libraries. This process will be described in detail in Section 4.2. Figure 3 shows the process diagram of the chosen implementation of the triode gain stage.

The grid conduction transfer function is described in Equation (1). It is adapted from a dynamic range compression curve [39] but computed from each sample instead of the RMS (Root Mean Square) or peak amplitude of the signal, making it a waveshaper. In Equation (1), t_0 represents the threshold, the point where grid conduction begins to alter the input signal. It is exposed as a parameter to the training algorithm, changing the span of the waveshaper's linear range. The motivation behind this waveshaping curve choice is that it does not completely clip the signal above it threshold. Instead it applies an attenuation to the positive side of the input signal mimicking the voltage drop occurring from grid current creating asymmetric distortion.

$$y[n] = \begin{cases} t_0 + \frac{x[n]-t_0}{4} & \text{if } 2(x[n]-t_0) > 1 \\ x[n] - \frac{3}{8}(x[n]-t_0+\frac{1}{2})^2 & \text{if } 2|x[n]-t_0| \leq 1 \\ x[n] & \text{otherwise} \end{cases} \quad (1)$$

The cathode bypass capacitor is modeled as an attenuating low-frequency shelving filter at 280Hz (a typical frequency taken from various schematics), with attenuation ranging from 0.0dB to –6.0dB set as a training parameter. Shelving filters attenuate or boost by a set gain all frequencies either above or below the cutoff frequency. For the tube cutoff distortion, a second waveshaper is applied after the polarity inversion. This waveshaper is a quadratic sigmoid curve altering only the positive half of the signal as is done by analogue tubes. The transfer function is given in Equation (2), with x_0 as the threshold, similar to the grid conduction, and also exposed as a parameter. Using a quadratic curve rather than the usual $\tanh$ makes it easier for CPUs to compute it and optimize it with SIMD instructions by not relying on the slower C or Julia standard library implementation.

$$y[x] = \begin{cases} \frac{-4}{(x_0-x[n]-2)^2} + x_0 + 1 & \text{if } x[n] > x_0 \\ x[n] & \text{otherwise} \end{cases} \quad (2)$$

The usual *gain* parameter on guitar amplifiers directly changes the attenuation between the first and second stages. A capacitor is often added across the *gain* control, called the *bright cap*, allowing high frequencies to bypass the voltage divider, thereby brightening the sound. Here it is modeled as a first order low shelving filter at 550Hz (as read from various circuit schematics) and with an attenuation depending on the first stage *gain* parameter. Finally, the signal passes through a low-pass filter for the plate capacitor, a high-pass filter for the coupling capacitor, and an output gain, each exposed as a parameter to the optimization algorithm.

The complete guitar amplifier implementation follows the diagram shown in Figure 1. It features four gain stages as described above, followed by the *tonestack* equalizer as modeled in [6], The power amplifier is modeled here as fully linear, although it is not in practice, and includes the *resonance* and *presence* filters. These act as low and high-boost shelving filters, respectively, and are found in many hardware models, they are set at 500Hz and 120Hz with the gains controlled by the parameters. The final stage in the guitar amplifier chain is the loudspeaker, represented by its impulse response and processed via convolution with the input signal.

Table 1 lists the parameters exposed to the optimization algorithm. All the filters used in the simulation are direct form IIR filters either second order for the parametric equalisers, and first order for all the others (lowpass, highpass, shelving). It is important to note that the model does not include any speaker cabinet response simulation usually done by convolution with its measured Impulse Response (IR). Guitar speakers have a complex frequency response and dramatically alter the sound of the complete guitar amplifier chain, adding this information to the already complex sound of a saturating guitar amplifier would not be valuable. Instead, the amplifier model is trained on targets that do not implement guitar speaker impulse response and add one later by convolution.

section	parameter name	range
gain stage 1-3	grid thresh	$[1, 3]$
	LSH gain	$[-6, 0]$ dB
	tube bias	$[0, 0.8]$
	tube linear range	$[0, 1]$
	LP freq	$[5000, 18000]$ Hz
	HP freq	$[15, 30]$ Hz
	gain	$[0, 1]$
gain stage 4	grid thresh	$[1, 3]$
	tube bias	$[0, 0.8]$
	tube linear range	$[0, 1]$
	LP freq	$[5000, 18000]$ Hz
	HP freq	$[15, 30]$ Hz
tonestack	low	$[0, 1]$
	mid	$[0, 1]$
	trebble	$[0, 1]$
other	input HP freq	$[70, 200]$ Hz
	presence HSH gain	$[0, 18]$ dB
	resonance HSH gain	$[0, 20]$ dB
	master volume	$[-60, 0]$ dB

Table 1. Model parameters, with 4 gain stages, it amounts to 33 parameters. *LSH/HSH gain*: Low SHelf/High SHelf filter gain, *LP/HP freq*: Low Pass/High Pass filter frequency.

4. TRAINING

The goal of the training process is to adjust the model's parameters to match the sound of a target guitar amplifier. Section 4.1 presents the design of the capture signal and the compound loss function, and Section 4.2 describes their integration into the Evolutionary Strategy scheme used to train the model. Finally, in Section 5, training results on a Neural Amp Modeler dataset are presented, and the accuracy of the resulting model is discussed.

4.1 Capture Signal and Loss Function

The signal capture method has been widely used in machine learning–based guitar amplifier modeling. In this work, we used capture signals from *Neural DSP* and *Neural Amp Modeler*, which are designed to match their respective training algorithms.

However, only portions of these signals were used, resulting in a capture signal of approximately 13 seconds, significantly shorter than those typically employed in this context (for reference, Neural DSP test signals are about one minute long, whereas Neural Amp Modeler uses signals of around three minutes). This substantially reduces the duration of the training phase.

Our capture signal is composed of:

- A one sample impulse, used to measure a short impulse response,
- Three frequency sweeps at different amplitudes for more precise impulse responses, taking into account the highly nonlinear behavior of guitar amplifiers
- Four stationary white noise segments at different amplitudes for another way of measuring amplitude dependent frequency response.
- Four sets of white noise pulses to measure the dynamic response of the model. The chosen envelope for the pulses is an exponential decay with a decay time of 500ms.

For fitness evaluation in the Evolutionary Strategy scheme discussed in Section 4.2, a loss function is designed based on the features we aim to adjust in our model with respect to the target model. The loss is not computed over the entire signal at once; rather, it is composed of several sub-losses, each operating on a specific segment of the capture signal, as described above. For the single-sample impulse, the loss is defined as in Equation (3), with P_w representing the Welch periodogram, MSE denoting the *Mean Square Error* function, $\hat{y}$ and y being respectively the estimate and ground truth signals..

$$\mathcal{L}_{dirac} = \text{MSE}(20\log_{10}(P_w(\hat{y})), 20\log_{10}(P_w(y))) \quad (3)$$

For the sweeps and stationary white noise, $\mathcal{L}_{sweep}$ and $\mathcal{L}_{noise}$ are defined as the MSE loss on the STFT spectrograms. For noise pulses, $\mathcal{L}_{pulses}$ is the MSE loss computed on the running RMS envelopes. This quantifies the amount of dynamic range compression a model applies to a signal and is particularly important in guitar distortion design. The final loss value is computed as shown in Equation (4).

$$\mathcal{L} = 15\mathcal{L}_{dirac} + 30(\mathcal{L}_{sweep} + \mathcal{L}_{noise}) + 10\mathcal{L}_{pulses} \quad (4)$$

The multiplicative constants are tuned to make the compound loss more sensitive to variations in selected aspects. Here, they are set so that the loss is particularly sensitive to changes in the two spectral losses of sweeps and stationary noise.

The final loss in Equation (4) was manually tuned after a long trial-and-error process.

4.2 Evolutionary Strategy

The purpose of the optimization is to tune the parameters of the model defined in Table 1 so that, given an input signal, it produces the same output as a target amplifier sound.

The chosen optimization algorithm is an Evolutionary Strategy called the $(\mu/\rho + \lambda)$ evolutionary strategy [40]. It belongs to the Evolutionary Algorithm category of optimization algorithms and is structured as follows. For each iteration (called a generation) of the algorithm, a set of μ solutions (a set of model parameters, called an individual) is defined. These are the parent individuals. The parents are then recombined in groups of ρ to form λ additional individuals (called the children). The $+$ in $(\mu/\rho + \lambda)$ indicates that parents are retained in the generation along with the children, unlike the $(\mu/\rho, \lambda)$ strategy, in which the parents are discarded.

Recombination can be performed in different ways; here, we take the average of the ρ contributing parents. After recombination, the child individual is mutated: random values are added to the parameters, drawn from individual Gaussian distributions and constrained by the defined range of each mutated parameter. The standard deviations $\sigma \in \mathbb{R}^N$ of these Gaussian distributions are adapted on every generation following what is called the *σ-self-adaptation*, described in [40] and presented in Equation (5). Here $\tau_0 = \sqrt{2N}^{-1}$ and $\tau = \sqrt{2\sqrt{N}}^{-1}$ are the learning parameters and $\mathcal{N}(0,1)$ is the Gaussian distribution of mean 0 and variance 1.

$$\sigma' = \sigma \exp(\tau_0 \mathcal{N}(0,1)) \cdot [\sigma_1 \exp(\tau \mathcal{N}(0,1)) \dots \sigma_N \exp(\tau \mathcal{N}(0,1))]^N \quad (5)$$

All parents and children are then evaluated by applying the parameters of an individual to a Faust DSP program, processing a signal, computing the *distance* of the output to that of the target amplifier using the loss function in Equation (4), and assigning a fitness value.

Finally, based on their respective fitness values, the best μ individuals out of the $\mu + \lambda$ are selected for the next generation. The algorithm repeats this process until convergence of the best fitness value in each iteration. Convergence is considered achieved after 50 iterations with a constant best fitness (meaning no better individual has been generated for 50 generations). The algorithm is implemented using the *Evolutionary.jl* library [41], available for the Julia programming language. The Faust compiler can directly generate Julia code, enabling integration into a scientific computing environment and allowing the use of Julia libraries such as *DSP.jl* [42], which may not be available when using C++.

5. DISCUSSION

5.1 Current Results

The dataset of target amplifier models was obtained from *Tone3000* [43], an online community database of *Neural Amp Modeler* models based on various real hardware amplifiers. A total of 94 models were tested, covering a variety of makes, models, and saturation levels, with no cabinet simulation (as explained in Section 3.2). All training processes were conducted on an Intel Ultra 7 165H CPU with 22 cores running at 4.7GHz with individual evaluation steps of the evolutionary process distributed across 8 threads. All audio is sampled at 48kHz without any resampling.

Table 2 lists some global statistics about the training process. The loss value has no unit or reference point, and its magnitude depends strongly on the scaling of each subloss. For the loss used in this training, shown in Equation (4), the first iteration with the model's parameters at their default values yields a loss value between 500 and 1000, while a loss of 0 would indicate a perfect match.

After training, the models are applied together with a guitar cabinet simulation to a raw electric guitar signal to perform a qualitative auditory evaluation. On evaluation, all trained models are close to their target model but there are still small audible differences. The two main aspects that can be improved are the amount of signal distortion (including both dynamic range compression and harmonic distortion) and the balance of the frequency content, particularly between low and high frequencies. For high-gain models, the distortion amount is similar, although it was evaluated qualitatively (a more rigorous approach will be discussed in Section 5.2). It is for lower-gain settings that the trained models deviate most from the target, both in loss values and listening tests. This is to be expected, as the trained model has four gain stages, whereas most low-gain or clean guitar amplifiers feature only two or three. These low-gain models were nevertheless kept in the dataset to evaluate how the system adapts to targets that do not correspond to its topology.

	iterations	sec./iter	Tot. time	final loss
min	54	3.02 s	239 s	39.74
max	249	5.59 s	1202 s	308.26
average	125.4	4.41 s	550 s	96.00

Table 2. Respective extrema values, average and standard deviation for number of iterations, duration of iteration (in seconds) and loss value over 94 amplifier emulation.

5.2 Limits and Further Improvements

In future training, a version of the distortion function with Anti-Derivative Anti-Aliasing (ADAA) [44] and upsampling will be implemented to improve aliasing rejection, at the cost of increased computation and longer training times. One hypothesis for the noticeable differences in low-frequency content is the presence of the attenuating low-frequency shelving filter, known as the *bright cap*, as described in Section 3. Tuning its attenuation or removing it altogether may yield better results for lower input-stage gain settings. The absence on nonlinear behaviour in the power amplifier simulation could also lead to significant differences in the situations where the target signal would involve power amplification saturation, adding various harmonic content depending on its topology (or class). Another hypothesis concerns the known poor resolution of the Fast Fourier Transform used in the loss functions for lower frequencies; finding a compromise to analyse and measure energy in the low-frequency range could allow for a more accurate estimation and a better match to the target.

As noted earlier in Section 5, the model performs worse for lower-gain targets because it has too many gain stages. Automatically recompiling the model with a different number of gain stages would allow a closer match to these targets. In future implementations, switching the number of gain stages during the optimization process will allow a better matching with less saturated settings. In the loss function, a useful addition would be the measurement of Total Harmonic Distortion (THD) as a means to capture the added harmonic content from heavy saturation. This measure would not focus on the positions of the added partials, but only on their amplitudes relative to the fundamental.

6. CONCLUSION

It demonstrates that rapid training can tune the proposed grey-box model to emulate a particular guitar amplifier, whereas this process was previously performed manually in the C++ plugin implementation presented in Section 3. They also show that evolutionary algorithms applied to the parameters of a Faust program yield promising results. Although some improvements to the grey-box model are still possible, our approach provides a viable alternative to computationally intensive neural networks. Moreover, since the optimized parameters being UI, human readable parameters, they can still be edited by hand for further fine tuning, learning, or sonic experiments. This gives a flexibility unavailable if using a neural network, where weights are fixed after training. Also, once the desired configuration and sound have been achieved, it is easy to fix the parameters in the Faust DSP code; the parts of the code relying on these parameters can then be computed at compile time, improving runtime performance and reducing user interface complexity.

Even if the results differ from training targets, they are still convincing guitar distortion sounds that can be used in a real-time music-making context. This article showed an automatic tone matching guitar amplifier DSP engine, trained on CPU by an Evolutionary Strategy algorithm. This method is not limited to guitar amplifiers, and in future works, will be applied to other systems, like modulation effects, filters and spatial effects.

7. REFERENCES

[1] S. Atkinson. Neural amp modeler. [Online]. Available: https://www.neuralampmodeler.com/

[2] Y. Orlarey, D. Fober, and S. Letz, "An Algebra for Block Diagram Languages," in *International Computer Music Conference*, ICMA, Ed., Gothenburg, Sweden, 2002, pp. 542–547.

[3] J. Pakarinen and D. T. Yeh, "A review of digital techniques for modeling vacuum-tube guitar amplifiers," *Computer Music Journal*, vol. 33, no. 2, pp. 85–100, 06 2009. [Online]. Available: https://doi.org/10.1162/comj.2009.33.2.85

[4] W. Pirkle, *Designing Audio Effect Plugins in C++: For AAX, AU, and VST3 with DSP Theory. Chapter 19 Addendum: Vacuum Tube & Distortion Emulation*. Routledge Taylor & Francis, 2019. [Online]. Available: https://www.willpirkle.com/fx-book-bonus-material/chapter-19-addendum/

[5] R. Giampiccolo, S. D'Angelo, A. Bernardini, and A. Sarti, "A quadric surface model of vacuum tubes for virtual analog applications," in *2023 26th International Conference on Digital Audio Effects (DAFx)*, 09 2023.

[6] D. T. Yeh and J. O. Smith, "Discretization of the '59 fender bassman tone stack," in *9th Conference on Digital Audio Effects (DAFx)*, 2006.

[7] L. W. Nagel and D. Pederson, "Spice (simulation program with integrated circuit emphasis)," University of California, Berkeley, Tech. Rep. UCB/ERL M382, Apr 1973. [Online]. Available: http://www2.eecs.berkeley.edu/Pubs/TechRpts/1973/22871.html

[8] I. Cohen and T. Hélie, "Simulation of a guitar amplifier stage for several triode models: examination of some relevant phenomena and choice of adapted numerical schemes," in *127th Convention of Audio Engineering Society*, New York, United States, Oct. 2009. [Online]. Available: https://hal.science/hal-00631757

[9] I. Cohen and T. Hélie, "Real-time simulation of a guitar power amplifier," in *13th Int. Conference on Digital Audio Effects (DAFx-10)*, Graz, Austria, Sep. 2010. [Online]. Available: https://hal.science/hal-00631752

[10] D. T. Yeh, J. S. Abel, and J. O. Smith, "Automated physical modeling of nonlinear audio circuits for real-time audio effects—part i: Theoretical development," *IEEE Transactions on Audio, Speech, and Language Processing*, vol. 18, no. 4, pp. 728–737, 2010.

[11] D. T. Yeh, "Automated physical modeling of nonlinear audio circuits for real-time audio effects—part ii: Bjt and vacuum tube examples," *IEEE Transactions on Audio, Speech, and Language Processing*, vol. 20, no. 4, pp. 1207–1216, 2012.

[12] D. T. Yeh and J. O. Smith, "Simulating guitar distortion circuits using wave digital and nonlinear state-space formulations," in *11th Conference on Digital Audio Effects (DAFx)*, 2008.

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

[14] J. Najnudel, "Power-Balanced Modeling of Nonlinear Electronic Components and Circuits for Audio Effects," Theses, Sorbonne Université, Jul. 2022. [Online]. Available: https://theses.hal.science/tel-03863726

[15] R. Müller, "Time-continuous power-balanced simulation of nonlinear audio circuits : realtime processing framework and aliasing rejection," Theses, Sorbonne Université, Jul. 2021. [Online]. Available: https://theses.hal.science/tel-03783502

[16] G. Golo, V. Talasila, A. van der Schaft, and B. Maschke, "Hamiltonian discretization of boundary control systems," *Automatica*, vol. 40, no. 5, pp. 757–771, 2004. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0005109804000147

[17] J. Engel, L. H. Hantrakul, C. Gu, and A. Roberts, "Ddsp: Differentiable digital signal processing," in *International Conference on Learning Representations*, 2020. [Online]. Available: https://openreview.net/forum?id=B1x1ma4tDr

[18] J. H. Engel, K. K. Agrawal, S. Chen, I. Gulrajani, C. Donahue, and A. Roberts, "Gansynth: Adversarial neural audio synthesis," in *7th International Conference on Learning Representations, ICLR 2019, New Orleans, LA, USA, May 6-9, 2019*. OpenReview.net, 2019. [Online]. Available: https://openreview.net/forum?id=H1xQVn09FX

[19] B. Hayes, J. Shier, G. Fazekas, A. McPherson, and C. Saitis, "A review of differentiable digital signal processing for music and speech synthesis," *Frontiers in Signal Processing*, vol. Volume 3 - 2023, 2024. [Online]. Available: https://www.frontiersin.org/journals/signal-processing/articles/10.3389/frsip.2023.1284100

[20] T. Hélie, "On the use of Volterra series for real-time simulations of weakly nonlinear analog audio devices: application to the Moog ladder filter," in *DAFx - International Conference On Digital Audio Effects*, Montréal, Canada, Sep. 2006, pp. 7–12, cote interne IRCAM: Helie06a. [Online]. Available: https://hal.science/hal-01106329

[21] C. Kemper, "Musical instrument with sound transducer," Patent EP1 883 064B1, 2006. [Online]. Available: https://patents.google.com/patent/EP1883064B1/en

[22] D. C. Borquez, E. Damskägg, A. Gotsopoulos, L. Juvela, and T. Sherson, "Neural modeler of audio systems," Patent EP3 828 878B1, 2023. [Online]. Available: https://patents.google.com/patent/EP3828878B1/en

[23] N. D. T. OY, "Emulating the behavior of audio systems," Patent DE202 022 100 831U1, 2022. [Online]. Available: https://patents.google.com/patent/DE202022100831U1/en

[24] "Nonlinear processor for audio signals," U.S. Patent US20 080 049 950A1, 2007. [Online]. Available: https://patents.google.com/patent/US20080049950A1/en

[25] R. Kuroki and T. Ito, "Digital audio signal processor with harmonics modification," JP Patent US5 841 875A, 1996. [Online]. Available: https://patents.google.com/patent/US5841875A/en

[26] M. Buffa and J. Lebrun, "Real time tube guitar amplifier simulation using WebAudio," in *Proceedings of 3rd Web Audio Conference, London, 2017.*, ser. Queen Mary Research Online (QMRO) repository. http://qmro.qmul.ac.uk/xmlui/handle/123456789/26089. London, United Kingdom: Queen Mary University of London, Aug. 2017. [Online]. Available: https://univ-cotedazur.hal.science/hal-01589229

[27] F. Eichas and Zölzer, "Virtual analog modeling of guitar amplifiers with wiener-hammerstein models," in *44th Annual Convention on Acoustics (DAGA)*, 2018.

[28] W. B. Langdon, R. Poli, N. F. McPhee, and J. R. Koza, *Genetic Programming: An Introduction and Tutorial, with a Survey of Techniques and Applications*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2008, pp. 927–1028. [Online]. Available: https://doi.org/10.1007/978-3-540-78293-3_22

[29] A. Horner, J. Beauchamp, and L. Haken, "Machine tongues xvi: Genetic algorithms and their application to fm matching synthesis," *Computer Music Journal*, vol. 17, no. 4, pp. 17–29, 1993. [Online]. Available: http://www.jstor.org/stable/3680541

[30] M. Cámara, Z. Xu, Y. Zong, J. L. Blanc, and J. D. Reiss, "Optimization techniques for a physical model of human vocalisation," in *26th Conference on Digital Audio Effects (DAFx)*, 2023.

[31] D. Etter, M. Hicks, and K. Cho, "Recursive adaptive filter design using an adaptive genetic algorithm," in *ICASSP '82. IEEE International Conference on Acoustics, Speech, and Signal Processing*, vol. 7, 1982, pp. 635–638.

[32] M. Macret and P. Pasquier, "Automatic design of sound synthesizers as pure data patches using coevolutionary mixed-typed cartesian genetic programming," in *Proceedings of the 2014 Annual Conference on Genetic and Evolutionary Computation*, ser. GECCO '14. New York, NY, USA: Association for Computing Machinery, 2014, p. 309–316. [Online]. Available: https://doi.org/10.1145/2576768.2598303

[33] C. Cooper, D. Murphy, D. Howard, and A. Tyrrell, "Singing synthesis with an evolved physical model," *IEEE Transactions on Audio, Speech, and Language Processing*, vol. 14, no. 4, pp. 1454–1461, 2006.

[34] M. Chemistruck, K. Marcolini, and W. Pirkle, "Generating Matrix Coefficients for Feedback Delay Networks Using Genetic Algorithm," *Journal of the Audio Engineering Society*, no. 8795, oct 2012.

[35] E. Ly and J. Villegas, "Genetic reverb: Synthesizing artificial reverberant fields via genetic algorithms," in *Artificial Intelligence in Music, Sound, Art and Design*, J. Romero, A. Ekárt, T. Martins, and J. Correia, Eds. Cham: Springer International Publishing, 2020, pp. 90–103.

[36] J. Coggin and W. Pirkle, "Automatic design of feedback delay network reverb parameters for impulse response matching," in *141st Convention of the Audio Engineering Society (AES)*, September 2016.

[37] B. Quiédeville. Ampmodeler, a guitar amplifier modeler plugin. [Online]. Available: https://github.com/BenjaminQuiedeville/AmpModeler

[38] J. Bezanson, A. Edelman, S. Karpinski, and V. B. Shah, "Julia: A fresh approach to numerical computing," *SIAM Review*, vol. 59, no. 1, pp. 65–98, 2017. [Online]. Available: https://epubs.siam.org/doi/10.1137/141000671

[39] D. Giannoulis, M. Massberg, and J. D. reiss, "Digital dynamic range compressor design—a tutorial and analysis," *Journal of the Audio Engineering Society (JAES)*, vol. 60, pp. 399–408, 2012.

[40] H.-G. Beyer and H.-P. Schwefel, "Evolution strategies – a comprehensive introduction," *Natural Computing*, vol. 1, pp. 3–52, 2002.

[41] A. Wilde, D. Thiem, Leo, A. Gupta, A. Haselgrove, D. Molina, Honeypot95, matago, Rogerluo, and V. Churavy, "Evolutionary.jl," jul 2021. [Online]. Available: https://doi.org/10.5281/zenodo.5110647

[42] S. Kornblith, M. Holters, G. Lynch, V. B. Shah, wheeheee, J. F. Santos, S. Russell, J. Kickliter, J. Bezanson, G. Adalsteinsson, A. Arslan, R. Yamamoto, jordancluts, kcin96, M. Pastell, E. Saba, T. Kelman, B. Arthur, T. Krauss, HDictus, H. El-Saawy, J. Kofron, E. Hanson, R. Luke, A. Kastanos, S. Verweij, S. Schoenbrod, and S. G. Johnson, "Juliadsp/dsp.jl: v0.8.4," jun 2025. [Online]. Available: https://doi.org/10.5281/zenodo.15632710

[43] Tone3000. [Online]. Available: https://tone3000.com

[44] S. Bilbao, F. Esqueda, J. D. Parker, and V. Välimäki, "Antiderivative antialiasing for memoryless nonlinearities," *IEEE Signal Processing Letters*, vol. 24, no. 7, pp. 1049–1053, 2017.