Digital Emulation of a Vacuum Tube Guitar Preamp

Taylor Zheng
Northwestern University
1700 Hinman Ave
Evanston, IL 60201
1-614-439-0928
TaylorZheng2015@u.northwestern.edu

Jixiao Ma
Northwestern University
1930 Ridge Ave
Evanston, IL 60201
1-847-868-6510
jixiaoma2014@u.northwestern.edu

Tae Hun Kim
Northwestern University
1637 Orrington Ave
Evanston, IL 60201
1-646-924-8407
taekim2013@u.northwestern.edu

ABSTRACT

We present a digital signal processing algorithm that takes an unprocessed electric guitar signal and outputs a processed signal that imitates how that particular guitar would sound being played through an overdriven vacuum tube guitar preamp.

General Terms

Algorithms, Design, Experimentation.

Keywords

vacuum tube, valve, emulation, duty cycle modulation, amplification, signal processing

1. INTRODUCTION

Vacuum tube guitar amplifiers are niche musical products which generate millions of dollars in annual sales. Vacuum tubes are an otherwise obsolete technology that is still being used in guitar amplifiers today because many electric guitarists prefer the way they distort and respond to playing dynamics compared to transistor amplifiers. The main reason why tubes sound different when they distort is the way they clip the signal, soft clipping the signal waveform instead of abruptly hard clipping (see figure 1).

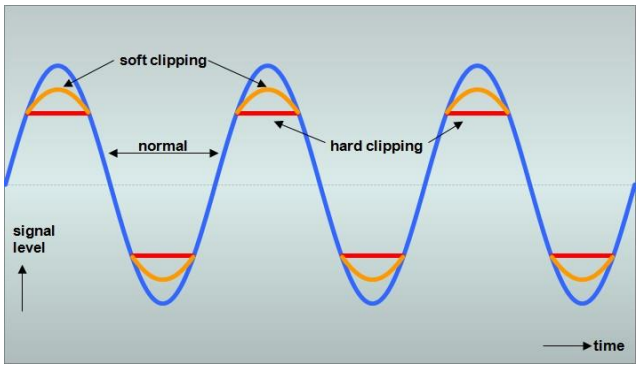

Figure 1: Soft vs. Hard Clipping

Digital modeling amplifiers already exist, but to many electric guitarists, they often fall short of capturing the unique sound of an overdriven tube amplifier. Our goal was to create an algorithm ableto emulate the cascaded soft clipping present in guitar vacuum tube preamps, given a suitable electric guitar signal encoded in an audio format.

In section 2 we present work already done in this area, while in section 3 we detail the process behind our algorithm. In section 4 we evaluate the performance of our algorithm, while in section 5 we detail future work that could be done.

2. RELATED WORK

Many major musical equipment manufacturers make digital modeling amplifiers that use algorithms to emulate guitar vacuum tube amplifiers. Examples include the Vypyr series produced by Peavey Electronics and the Mustang series produced by Fender Musical Instruments Company. Other companies make dedicated digital effects processors that also have vacuum tube amplifier emulations. Examples include the RP series produced by Digitech and the POD series produced by Line6.

There also exist VST plug-ins for applications such as Audacity that emulate tube preamps, such as Voxengo: Tube Amp, Silverspike: Ruby Tube, and Togu Audio Line: TAL Tube. The specific VST plug-in we will be using for comparison purposes in section 4 is Nick Crow Lab: Tube Driver.

3. DESIGN

3.1 Individual Gain Stages

The algorithm accepts a .MP3 music file of a guitar signal as input. Other music formats supported by MATLAB's audioread function will work as well. The algorithm for each simulated gain stage is based on how a vacuum tube triode gain stage functions as detailed in [1]. It first applies a low pass filter to the input to simulate the low pass filter created by Cf, Rg (see Figure 2), and the vacuum tube's capacitance, with the -3 dB frequency calculated by the formula:

$$Hz = \frac{1}{2\pi \times Rg \times (Cf + \text{Tube Capacitance})}$$

Next, the algorithm applies a high pass filter to simulate the high pass filter created by Rk and Ck (see figure 2), with the -3 dB frequency calculated by the formula:

$$Hz = \frac{1}{2\pi \times Rk \times Ck}$$

It then amplifies the signal slightly, soft clips the signal on one side of the waveform, and flips the signal upside down to simulate how a vacuum tube amplifies and clips a signal. The soft clip transfer function is represented by:

$$f(x) = y + \frac{\log(x - y + 1)}{\log(20)}$$

for signals greater than the desired clipping threshold, where x is the input signal and y is the desired clipping threshold between 0 and 1. Parts of the signal that are still greater than 1 after applying this function are set to equal 1.

The algorithm then centers the waveform by filtering out the DC offset in order to simulate how Co filters out DC offset. The values of Co and Rl are simulated as being large enough to not form a high pass filter with respect to audio frequencies, thus having no effect on our algorithm.

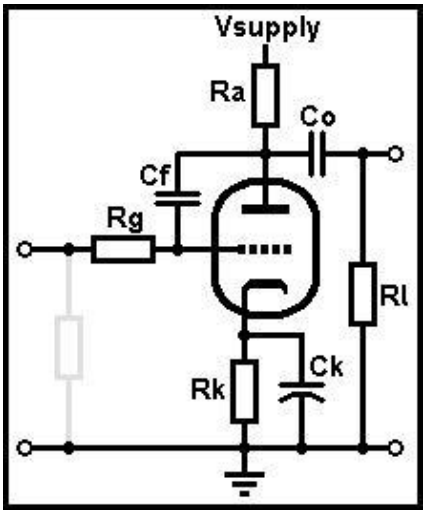

Figure 2: Typical Triode Vacuum Tube Gain Stage

3.2 Overall Preamp Structure

Our simulated guitar vacuum tube preamp has 3 individual tube gain stages in order to emulate the cascading gain stage designs of many modern guitar vacuum tube preamps. Using more than one clipping stage results in duty cycle modulation, which is a shifting of the average DC offset of the waveform based on the input signal strength. This generates mostly odd-order harmonics with some even-order harmonics which increase as input signal gain increases.

The Graphical User Interface (GUI) for our algorithm is based on those of VST plug-ins for Audacity as mentioned in the previous section. Each gain stage has an individual gain knob (see Figure 3), and there are treble, mid, and bass knobs in the EQ section to simulate the tone stack, which is the equivalent of a passive EQ section in a guitar amplifier. The simulated tone stacks are based off of those given by the stock values in the Duncan Tone Stack Calculator, downloadable at the following web page: http://www.duncanamps.com/tsc/

There are 3 different EQ voicings available, which shift the frequency responses of the filters in the EQ section. If the Fender voicing is chosen, the tone stack is moved in between stages 1 and 2, while if the Vox or Marshall voicings are chosen, the tone stack is moved to immediately after stage 3. There are also 8 presets that simulate the frequency responses of different types of speakers available from Jensen Loudspeakers, specifically: the Mod 6-15, the P8R, the C8R, the P10R, the C10R, the Blackbird 12, the P12N, and the C12N. At the very end of the signal chain, we scale the signal so that the maximum of the absolute value of the signal vector is 1, then multiply the signal by a value between 0 and 1 that is determined by where the output volume control is set in the GUI.

Overall, the signal chain is ordered as such: *Gain stage 1 => (Tone stack if Fender preset chosen) => Gain stage 2 => Gain stage 3 => (Tone stack if Vox or Marshall preset chosen) => Set of filters to emulate speaker => Output volume scaling.*

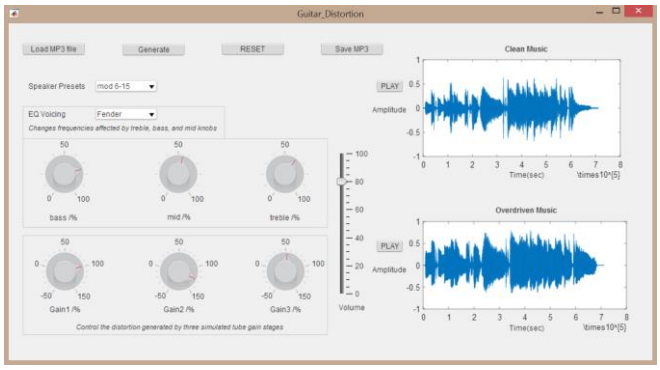

Figure 3: GUI to Control Algorithm Parameters

4. PERFORMANCE

To test the accuracy of our entire system, we used Jensen Loudspeakers' sound sample files collection, located at: http://www.jensentone.com/sound_sample_files. The collection contains the same electric guitar passage played through a vacuum tube amplifier set clean and then overdriven, all through a variety of different speakers. This allows us to process the clean version of the passage through our algorithm and compare the output to the overdriven version., which we treated as our ground truth.

For testing purposes, we decided to use Mel Frequency Cepstral Coefficients (MFCC) to measure the difference between our processed sound sample and the overdriven sample. To do this, we used the Rastamat package for Matlab provided at http://labrosa.ee.columbia.edu/matlab/rastamat/ to generate MFCC plots of our processed sample and the overdriven sample; then, since the plots can be represented as matrices, we subtracted one plot from the other and took the absolute value of the result. To minimize the resulting value, we used the patternsearch function in Matlab, a type of hill-climbing algorithm. Given the range of values each setting can achieve in our GUI as a constraint, the minimum value for our algorithm was approximately 116,000; however, when these constraints were lifted, the global minimum became approximately 75,000. For comparison's sake, we repeated this process for a version of the clean sample processed through the Nick Crow Lab: Tube Driver VST mentioned in section 2 and received a value of approximately 100,000, given the settings the VST can achieve as a constraint. Thus, our algorithm achieves comparable performance to existing VST algorithms and can outperform them if there are no constraints on the arguments passed to our algorithm; however, we did not allow our GUI to pass such arguments, because the values taken would have been impractical due to their sheer maginitude.

5. FUTURE WORK

Our thanks to ACM SIGCHI for allowing us to modify templates they had developed.

6. REFERENCES

[1] Bowman, M., Debray, S. K., and Peterson, L. L. 1993. Reasoning about naming systems. *ACM Trans. Program. Lang. Syst.* 15, 5 (Nov. 1993), 795-825. DOI= http://doi.acm.org/10.1145/161468.16147.

[2] Ding, W. and Marchionini, G. 1997. *A Study on Video Browsing Strategies*. Technical Report. University of Maryland at College Park.

[3] Fröhlich, B. and Plate, J. 2000. The cubic mouse: a new device for three-dimensional input. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems* (The Hague, The Netherlands, April 01 - 06, 2000). CHI '00. ACM, New York, NY, 526-531. DOI= http://doi.acm.org/10.1145/332040.332491.

[4] Tavel, P. 2007. *Modeling and Simulation Design*. AK Peters Ltd., Natick, MA.

[5] Sannella, M. J. 1994. *Constraint Satisfaction and Debugging for Interactive User Interfaces*. Doctoral Thesis. UMI Order Number: UMI Order No. GAX95-09398., University of Washington.