Sound Devices for Musicians

Jack Cassidy - Computer Engineer
Benjamin Mullin - Cyber Security Engineer
Bradley McClellan - Software Engineer
Harry Burnick - Electrical Engineer
Ian Bixler - Electrical Engineer
Julia Kroeper - Electrical Engineer
Client- Randall Geiger

Vacuum Tubes and Audio Amplification

- Originally developed at the beginning of the 20th century
- Became ubiquitous in audio amplification by the end of the 1920s
- Slowly phased out in the mid 1970s - 80s due to solid-state transistors
- Downfall was due to costly fragile tubes versus cheap reliable solid state components

The Overarching Goals for Our Project

1. Learn about the distortion characteristics introduced by vacuum tubes & use that knowledge to implement an accurate emulation.
    a. What makes a tube sound like a tube?
2. Design a system that can accurately replicate these distortion characteristics on any given audio signal.
    a. Allow the user to add distortion to any type of audio input they might want
3. Add in user-adjustable parameters to allow for custom distortion factors
4. User acceptance testing to see if the general public enjoy the distortion

Detailed Design

1. **Develop a Mathematical model to simulate the distortion created by a Vacuum tube.**

- Obtain tube amplifier for testing.
- Perform Spectral analysis using different audio signals.
- Background Research

2. **Create a program to apply said Mathematical model to a signal.**

- Python
- Max 8
- Implement and Test Mathematical model.

3. **Allow the user to edit the Mathematical Model in real time**

- Perform real time audio analysis

4. **Test our creation**

- Survey our creation with people with differing levels of musical knowledge

Work Progress

- Created a nonlinear transfer function to apply to audio signal

$$f = (\frac{(3 + A) \times sin(Cx - \phi)}{2})(1 - \frac{sin^2(Cx - \phi)}{3})$$

- Created a program to apply created formula to different audio clips and inputs
- Began User Acceptance Testing to see how different forms of distortion are received and how genre can affect this result

A = gain
C = compression
φ = phase
x = audio signal

User Acceptance Testing

One important part of our project is deciding which type of distortion people prefer. To accomplish this task, we have decided to conduct Self-Administered Trials to collect such data. We've created a Google Form for this purpose. Initially, the form asks questions about the audio device and the environment in which the test is being taken. This allows us to control for factors that could contribute to faulty data.

Testing Procedure

The following will be carried out 20 times with different audio clips.

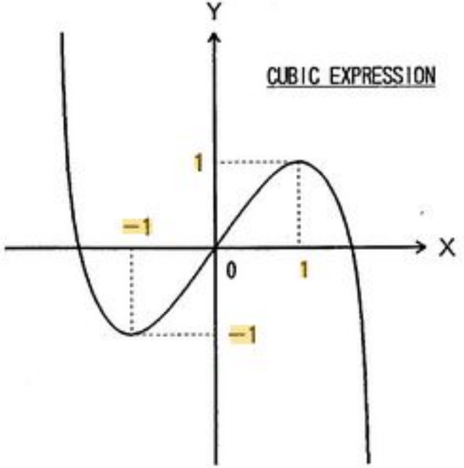
Trial Subject listens to two provided audio files (one with and one without distortion applied) and selects one of the three following options:

- First sample was better
- Second sample was better
- No difference between samples

Nonlinear Transfer Function

- Yamaha engineers Toshinori Araya and Akio Suyama in 1996
  - Digitally applying a distortion effect to an audio signal without clipping
  - Characteristics of vacuum tube amplifiers

$$y = \frac{3}{2}x(1 - \frac{1}{3}x^2)$$



F I G. 4

[1] T. Araya and A. Suyama, “Sound Effector Capable of Imparting Plural Sound Effects Like Distortion And Other Effects,” Oct. 29, 1996

Nonlinear Transfer Function

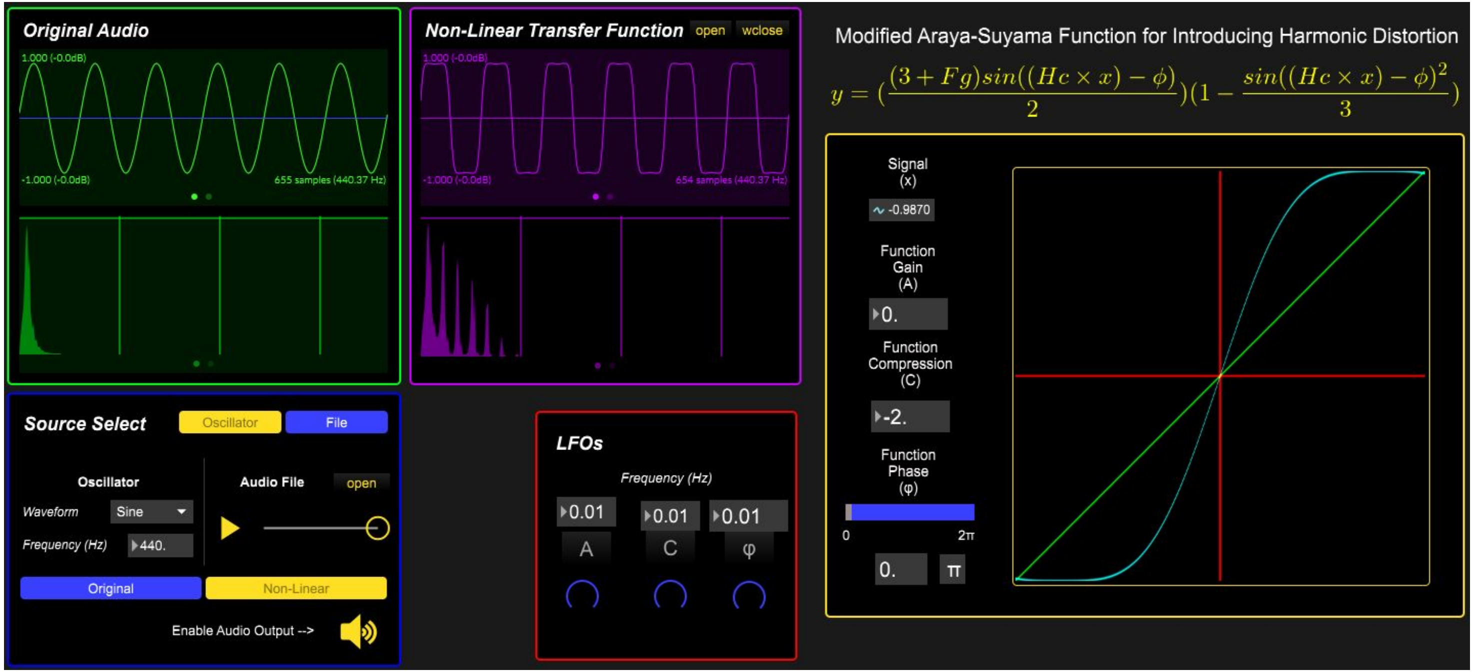
Python Prototype

```python
from scipy.io import wavfile
import numpy as np

# Read the WAV file

input_file = input("Input file: ")
sample_rate, audio_data = wavfile.read(input_file)

# Define the non-linear response curve function
def nonlinear_function(x):
    return (3 * x / 2) * (1 - (x ** 2) / 3)

# Normalize audio data to the range [-1, 1]
audio_data_normalized = audio_data.astype(np.float32) / np.max(np.abs(audio_data))

# Apply the non-linear function
audio_data_transformed = nonlinear_function(audio_data_normalized)

# Scale the transformed data back to the original range
transformed_audio = np.int16(audio_data_transformed * np.iinfo(np.int16).max)

# Save the processed audio to an output file
wavfile.write('output.wav', sample_rate, transformed_audio)
```

- Any polynomial function can be used
- Changing different parameters of the equation to observe spectral response
    - Odd-order functions result in only odd-order harmonics and vice-versa
- Only works on audio files

Nonlinear Transfer Function

Realtime Processing with Max 8

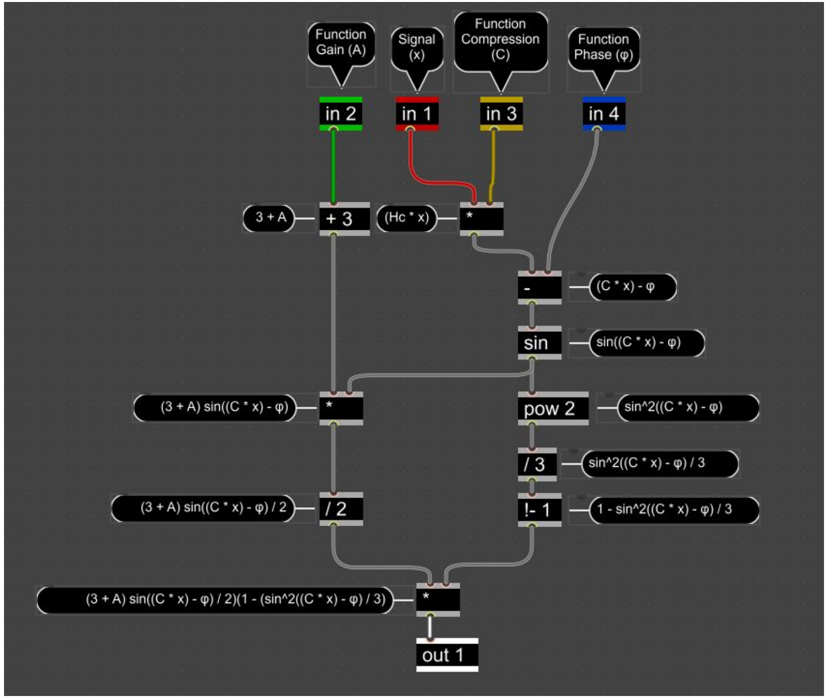


Nonlinear Transfer Function

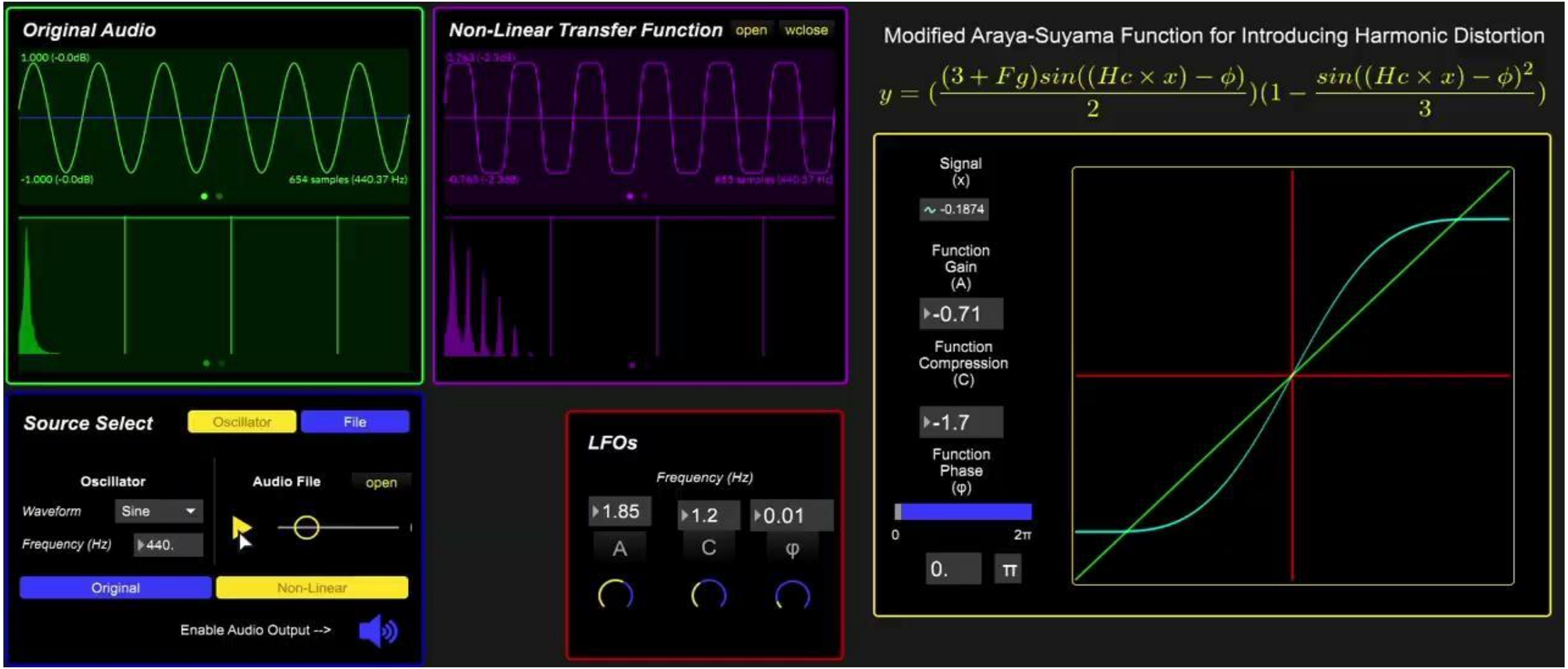
$$f = \left(\frac{(3 + A) \times sin(Cx - \phi)}{2}\right)\left(1 - \frac{sin^2(Cx - \phi)}{3}\right)$$

Realtime Processing with Max 8

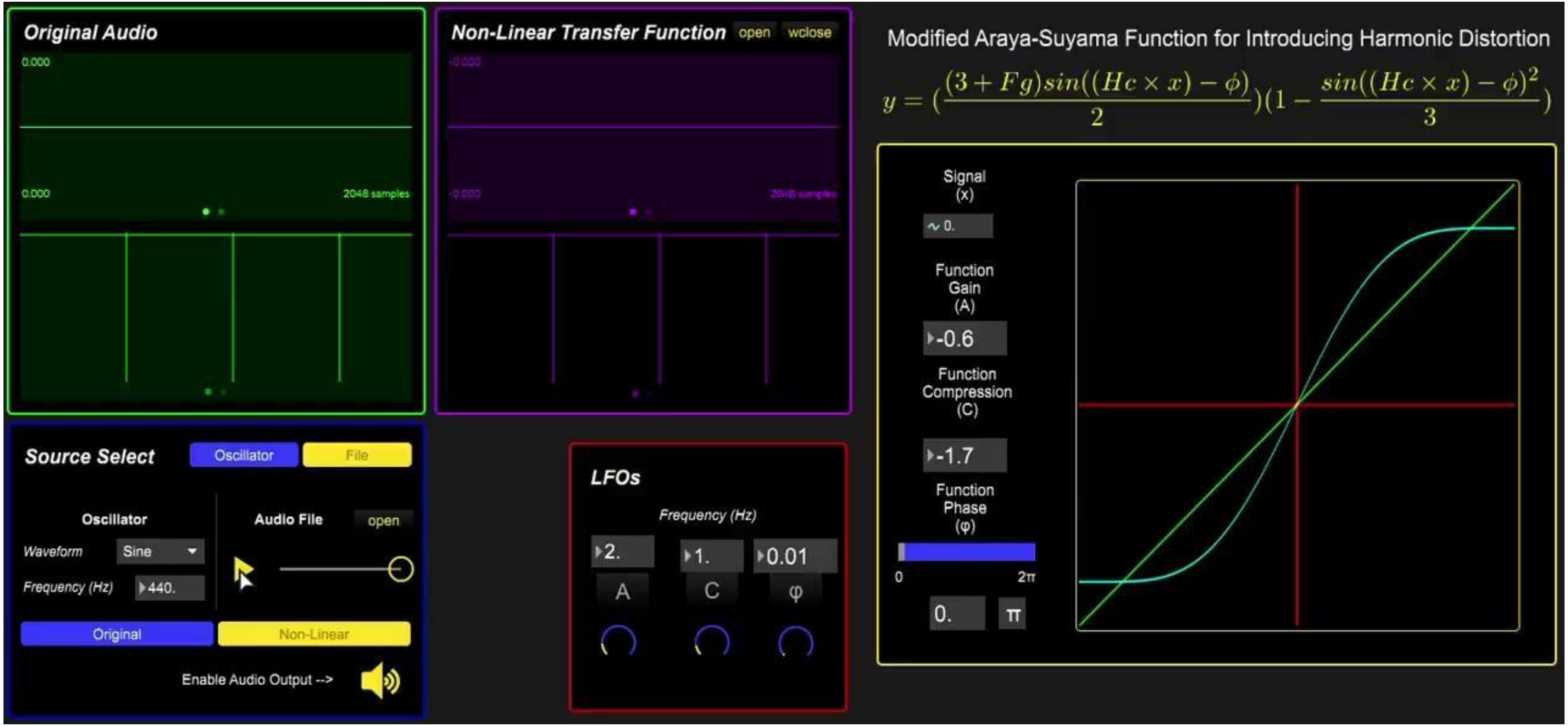
- Max 8 is a visual programming environment for MIDI, audio, and video processing
- Allows testing different parameters in real time on audio clips and test signals



Demo

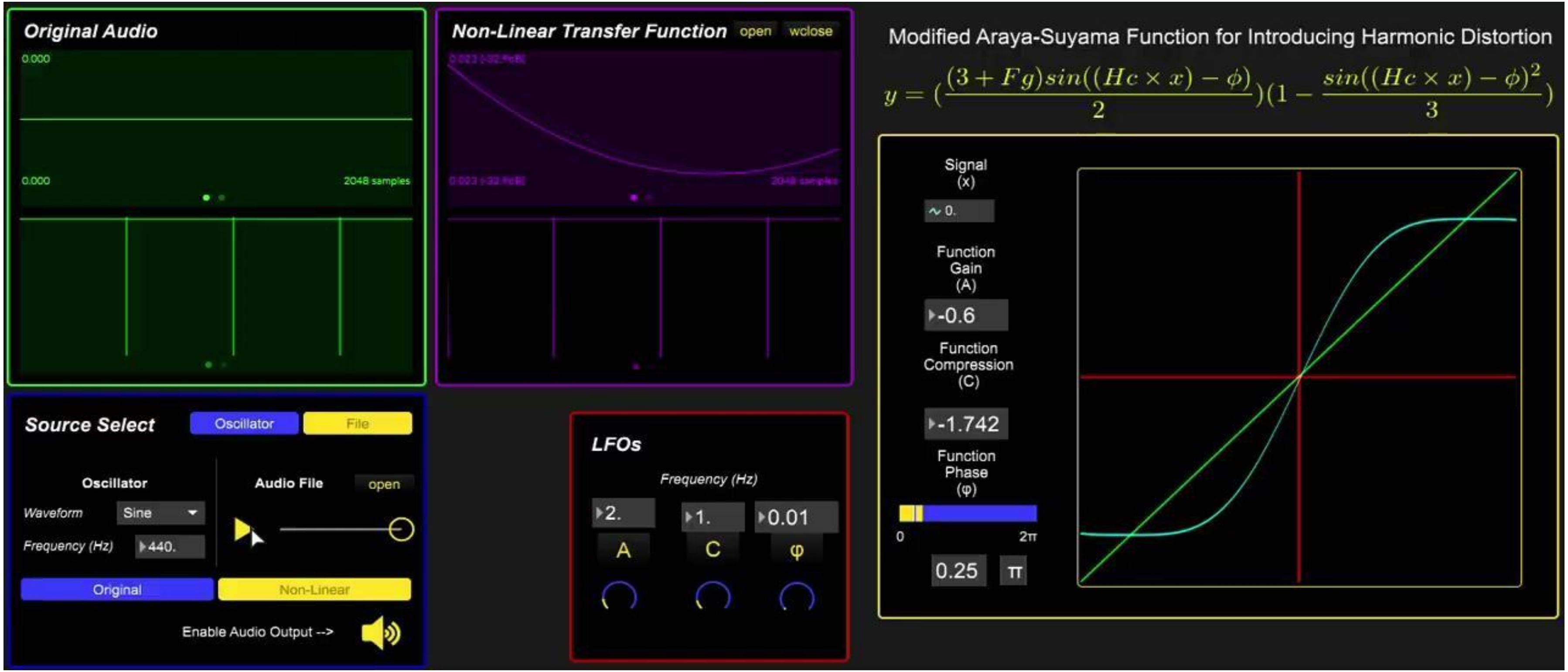

Original Audio
1.000 (-0.0dB)
-1.000 (-0.0dB)
654 samples (440.37 Hz)
Non-Linear Transfer Function
open
wclose
Modified Araya-Suyama Function for Introducing Harmonic Distortion
$y = (\frac{(3+Fg)sin((Hc \times x) - \phi)}{2})(1 - \frac{sin((Hc \times x) - \phi)^2}{3})$
Signal
(x)
-0.1874
Function
Gain
(A)
-0.71
Function
Compression
(C)
-1.7
Function
Phase
(φ)
0
2π
0.
π
Source Select
Oscillator
File
Oscillator
Waveform
Sine
Frequency (Hz)
440.
Audio File
open
Original
Non-Linear
Enable Audio Output -->
LFOs
Frequency (Hz)
1.85
1.2
0.01
A
C
φ

Demo



Original Audio
0.000
0.000
2048 samples
Non-Linear Transfer Function
open
wclose
Modified Araya-Suyama Function for Introducing Harmonic Distortion
$y = (\frac{(3+Fg)sin((Hc \times x) - \phi)}{2})(1 - \frac{sin((Hc \times x) - \phi)^2}{3})$
Signal (x)
0.
Function Gain (A)
-0.6
Function Compression (C)
-1.7
Function Phase (φ)
0
2π
0.
π
Source Select
Oscillator
File
Oscillator
Waveform
Sine
Frequency (Hz)
440.
Audio File
open
Original
Non-Linear
Enable Audio Output -->
LFOs
Frequency (Hz)
2.
1.
0.01
A
C
φ

Demo


Original Audio
0.000
0.000
2048 samples
Non-Linear Transfer Function
open
wclose
Modified Araya-Suyama Function for Introducing Harmonic Distortion
$y = (\frac{(3+Fg)sin((Hc \times x) - \phi)}{2})(1 - \frac{sin((Hc \times x) - \phi)^2}{3})$
Signal (x)
0.
Function Gain (A)
-0.6
Function Compression (C)
-1.742
Function Phase (φ)
0
2π
0.25
π
Source Select
Oscillator
File
Oscillator
Waveform
Sine
Frequency (Hz)
440.
Audio File
open
Original
Non-Linear
Enable Audio Output -->
LFOs
Frequency (Hz)
2.
1.
0.01
A
C
φ

Challenges and Solutions

- Real-Time Audio Processing could not be done first software implementation of our created formula
    - Switched from Matlab to MaxxAudio
- Real time user-customizable effects
    - Creating a program that could allow the individual to change the distortion formula in real time, while also being able to choose from a selection of presets
    - Had to add to our software program so that it could take in inputs in real time and change the distortion formula
- Mathematical Model
    - Had to create a mathematical model that could simulate a tube amp, while also allowing a third party to change the model in real time
    - Adapted a previous tube amp model, and by adding different inputs, allowed the user to change what the distortion sounded like

Conclusion

- Currently we have made major strides in creating a model to add distortion to audio in real time, and implementing it in software.
- We are currently in the stage where we are asking for feedback for different distortion types for added presets with our final project
- After that all that is left is to assemble our completed project with these different presets and the ability for a third party to customize the distortion formula in real time