Cardiff University

CM3106 Multimedia

# MATLAB Interactive Fourier-based Synthesiser

*Student Name:*
Geraint Harries
*Student Number:*
1100682

*Lecturer(s):*
Prof. David Marshall
Dr. Kirill Sidorov

December 6, 2013

# 1 Program Overview

This program is inspired by a piece of audio software called Iris by Izotope Inc. I have used this software as inspiration. I am able to load audio and graphically represent and edit the time-frequency distrubution. I am also able to apply other audio effects such as Tromolo, Ring Modulation, Wah Wah etc as well as volume shaping. Many of the variables for each audio effect are modifiable by the user. Below is a description of the basic algorithm design and main elements of each section of my program. There are also detailed comments within the code.

# 2 Algorithm List and Description

## 2.1 Reading in an audio input file

This function allows the user to select an audio file from thier directory and save the frequency and file as global variables WAVEFREQUENCY and WAVEFILE respectivly.

## 2.2 Compute Fourier

This function computes and shows the short term fourier transform of an audiofile.

The Pseudocode:

- Get short term fourier transform of the audio file.
- Convert the complex magnitude of the fourier transform into absolute values.
- Show the output and save to a variable.

## 2.3 Volume Shaping

This function generates a basic volume shaping on an audio file. It does so by generating a graph of a basic ADSR shape (Shown in Figure 1) and then calculating the dot product of the ADSR and audio file.

The Pseudocode:

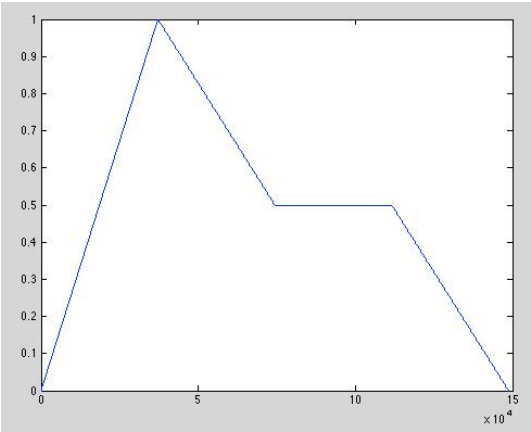


Figure 1: ADSR Schematic

- Generate the ADSR graph.
- Calculate the dot product of the ADSR graph and the audio file.
- Return calculated value

## 2.4 Reverb

### 2.4.1 Standard Reverb

This function generate a reverb effect on given audio given a users parameters. It adds this vector to the original audio file and then returns it.

The Pseudocode:

- Generate a reverb vector by using the users gain and delay value.
- The vector is added to the original audio file.
- Assign the calcuation to the audio file.

### 2.4.2 Reverb Convolution

This function generates a reverb effect on given audio using another audio file the user has selected. It convolves the two audio files and normalizes the

output to +-1.
The Pseudocode:

- Find smallest of two files
- Fast fourier transform both
- Calculate the dot product
- Inverse fast fourier transform the calculated dot product
- Return the first N (length of shortest) elements

## 2.5 Wah Wah

This function runs the wah wah effect on the audio file. It does so by creating a variable peak response of a filter and applying it to the audio file.

The Pseudocode:

- Create empty vector same size as the audio file
- Iterating through each element in the vector calculate the value for the oscilating sine wave and add or remove it from the from the original file.
- Add it to the empty vector.

## 2.6 Ring Modulation

This function allows the user to generate a ring modulation to their audio file. A carrier vector is created and then a dot product is calculated.

The Pseudocode:

- Create a carier vector
- Calculate the dot product of the carrier and audio file.
- Assign dot product calculation to original audio file.

## 2.7 Phase Vocoder

I have implemented two options for the user regarding phase vocoding. One uses user entered values, the user is able to specify the ratio by which the pitch increases (See Figure 2). The other is by using the GUI keyboard (See Figure 3). Both ways use the same algorithm. It generates a slower or faster file given a fraction (either the user supplies or which is pre-set). The audio file is then resampled using the same ratio. The ratios were found from url-http://oeis.org/DUNNE/TEMPERAMENT.HTML

The Pseudocode:

- Generate ratio using either preset or user given numerator and denominator.
- Generate a changed speed audio file given the ratio
- Resample the speed changed audio to the desired pitch given the ratio

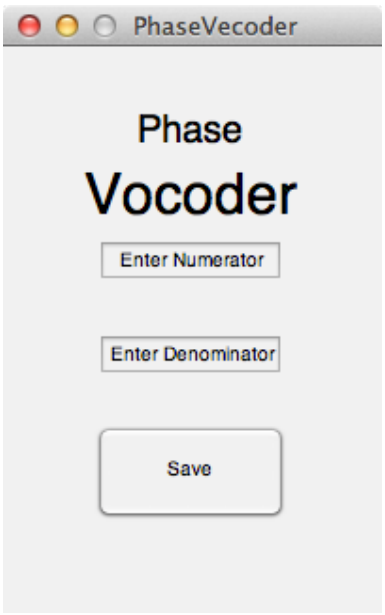


Figure 2: User input for phase vocoder

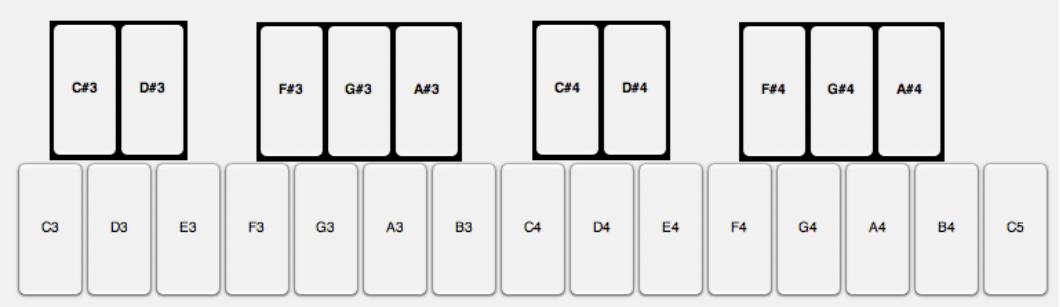


Figure 3: Keyboard with specific phase vecoded frequencies

## 2.8 Save

This function saves the graphically audio file. The edit function does not save the file after editing as to allow the user the choice of whether to edit their change. It simply uses istft on the edited spectrogram and saves that to the global variable of the audio file.

## 2.9 Edit

This function creates a mask with the same dimensions as the fourier transform of the audio file. Using imfreehand, it allows the user to draw on the transparent mask and then generates a binary image of the users drawing. A dot product is then generated of the binary image and spectrogram. The changed spectrogram is then displayed. N.B. The new spectrogram is not saved, it only saves if you press the save button.

The Pseudocode:

- Create and display transparent mask of same dimensions as fourier transform of audio file.
- Record the free hand drawing of the user
- Generate a binary mask from the drawing
- Calculate the dot product of the binary mask and fourier of original audio file.

### 2.10 Tremolo

This function generates a tremolo effect given the users values for alpha and FC. It generates a tremolo vector and then calculates the dot product for the vector and original audio file.

- Generate tremolo vector given user input of alpha and Fc
- Calculate the dot product of vector and original audio file
- Assign to original audio file

### 2.11 AudioPlayer

I have implemented audioplayer which is able to do the following:

- Play
- Pause
- Stop
- Record

### 2.12 Loop

I implemented looping by creating an infinite loop and outputting the file within it

### 2.13 Reverse

Using the flipud() MATLAB function, I was able to flip the audio file matrix. This essentially reversed the audio.

## 3 Novel Features

As previously mentioned in subsections 2.42, 2.11, 2.12 and 2.13, I implemented reverb convolution, recording, looping and reversing. These are my novel features as they are not strictly part of the core criteria (although reverb is, reverb convolution isn't). You wil see the detail of these features

in the aforementioned subsections. There are also detailed comments within the code.

# 4 Satisfying the basic criteria

I beleive that I satisfied all of the basic criteria of the coursework. I implemented:

- A method of reading in an input audio file (2.1)
- Computed its underlying time-frequency distribution (2.2)
- Provided an interactive means of editing this via its displayed spectrogram (2.8)
- The resulting audio is able to be played back (2.11, 2.7)
- New sounds are able to be generated using the interactive keyboard. (2.7)
- Volume Shaping to control the basic sounds produced (2.3)
- Tremolo (2.10)
- Wah Wah (2.5)
- Ring Modulation (2.6)
- Basic Reverb (2.4.1)

Given these functions I feel I have met the core requirements. I have also implemented

- Reverb Convolution (2.4.2)
- The ability to record (2.11)
- The ability to loop audio (2.12)
- The ability to reverse audio. (2.13)

These features were not part of the core requirement and therefor count as novel feature. Therefore, in that regard, I feel that I have meet all the criteria of this coursework including novel features.