# Vocal Synthesizer & Harmonizer

Rhiannon O'Brien and Finn Denton

MUMT 307 Final Project
Dr. Gary Scavone

## 1. Project Summary (Written by: Finn)

Our goal was to implement an interactive MATLAB script that allows a user to record a sample, then manipulate it with different harmonies and effects. We were motivated by the euphonious nature of harmonics, as well as the technical implementations we have learned in class. We wanted to make manipulation more accessible and user-friendly for those with less technical experience so that this project could serve as both an educational and experimental tool. By simplifying the concepts of harmonics and effects, the user is able to see the direct causations of different decisions in real time.

With the use of displayed triads, the user is able to see how different combinations of harmonies affect the sample differently: some sounding more pleasant than others. Aside from its use as a playground for sound effects, we also wanted this project to serve as a way to test pitch inputs from the user. We aimed to implement a pitch detector and estimator that detects the input and rounds it to the closest note. This is used to manipulate a more sonically distinct sound, but also to confirm to the user which note was inputted. From this, the user can either read from the few provided triads, or apply their own music knowledge to layer harmonies. With a clear framework and UI, we aimed to take something complex that we learned, and streamline it for more widespread use.

## 2. Methodology

### 2.1. Pitch Estimation (Written by: Rhiannon)

The first goal was to estimate the pitch of the sound recording made by the user. There were a few options we considered, one of which was the pitch() function in MATLAB. We were unsure about the accuracy of this function, which Dr. Scavone confirmed for us, that the function may not always produce a correct or accurate result. Instead, he pointed us to YIN, a fundamental frequency estimator. A GitHub repository done by Orchisama Das implemented this estimator in a MATLAB script. Using this script allowed us to focus on the harmonizer and addition of sound effects, rather than creating a pitch estimator ourselves. The YIN pitch estimator is implemented by calculating a difference function, then calculating the cumulative mean normalized difference function using the previously calculated difference function to reduce errors. The next step is to set an absolute threshold to further reduce errors. Parabolic interpolation follows, which is a solution if the period is not a multiple of the sampling period. Local estimation for accuracy comes from the use of very small time frames/samples. The YIN algorithm works to make an accurate and local pitch estimation. The pitch estimation is given in samples per second, so it must be converted into Hertz to see the true frequency. After trial and

error and research, we realized that being able to create our harmonizer would become complicated using the original voice recording. This is a result of a few factors, such as inaccuracy of user pitch, background noise, clipping in the recording/technical issues, and pitch shifting in general, with the low-quality recording. Considering these factors, to make a clean harmonizer, we decided to convert the original voice recording to a pure tone. This would allow us to reproduce accurate interval pitches that would be simpler to work with and sound more accurate.

### 2.2. Harmonizing (Written by: Rhiannon)

The harmonizer portion of this project was the focus of our original idea. Our aim was to be able to produce different intervals stemming from the user-sung note to play for the user, layered on their original sound. Using the pure tone, we multiplied said wave by the associated ratio of the interval we wanted to produce. For example, the ratio to create a Major 3rd is 5/4, and the ratio to create a Perfect 5th is 3/2. To apply the interval, we multiplied the ratio by the original pure tone and created a new sine wave. To play the original tone and the tone multiplied by the interval ratio, we added the new tone to the original tone and divided it by two. Our program allows the user to add multiple intervals before playing the created chord, so the addition will repeat to add the new tone into the combined signal. Using a formula to find the MIDI note number associated with a certain frequency (assuming equal tuning at 440 Hz), we added a portion of the script that would display to the user what chords they could create with their sung note as the root, including the intervals to use and the MIDI note numbers.

### 2.3. Effects (Written by: Finn)

After completing the harmonic aspect of our script, our curiosity led us to explore different effects that we could implement. Through some experimentation with the original pure tone sample, we found that we were more limited with the audible effects to implement. Because we were using a shortened, pure toned version, effects like echo and reverb were not as clear. Based on this, we chose the following effects to implement: phaser, ring modulator, distortion, chorus, and vibrato. In the end, since we switched from the pure tone to the original sample, we did not need to limit ourselves as much, but this is what originally guided us. Below are the effects, along our parameters and modes of implementation:

A. Phaser
    - Parameter(s): 0.2 Hz sweep rate, 0.7 depth, 800 Hz base cutoff, and Q = 4 resonance, Mix: 70% dry/wet.
    - Uses a time-varying all-pass filter with a low frequency oscillator to create a sweeping effect.
    - Sweeps sinusoidally creating specific phase cancellations in the sample.
B. Ring Modulator
    - Parameter(s): 40 Hz modulation frequency, Mix: 70% wet/dry.
    - Creates metallic sound with modulation of carrier frequency and original sample.
    - After multiplying, it mixes the result with the original sound using wet/dry balance.
C. Distortion

- Parameter(s): Clip level of 0.5
- Takes maximum of the following to set to sound: negative clip level and minimum of the clip_level and input sound

D. Chorus
- Parameter(s): 20 ms delay, 10 depth, 1.5 Hz LFO rate, Mix: 50% wet/dry
- Indexes through original sample and adds LFO to each interval of original sound.
- Uses sound from delay-line and adds wet mix times new sound to the original sound.

E. Vibrato
- Parameter(s): 5 Hz rate, 3 depth, Mix: 80% wet/dry
- Creates time varying delay to create pitch oscillation by multiplying the depth samples to a sine wave
- Uses linear interpolation in between samples to preserve timbre of voice while also applying vibrato.

Each effect we implemented allows the user to playback their original sample side by side with the manipulated sound. As the user enters the dropdown menu, they can apply different effects to the same original sample, without applying permanent changes on the sample. This was implemented with the intention of ease for the user, to allow room for error or typos.

## 3. Challenges & Solutions: Finn

Throughout the project, our idea evolved and became more refined through several rounds of trial and error. The first challenge I came across was during the implementation of the layering of harmonies. As I would play back the harmonized sound after the original sample, I noticed some clipping and changes in volume that did not align with the intended layering. However, through some research, I found that because I was adding to the sample during the harmony layering, that it exceeded +-1 in amplitude. I took this, and also applied it to the effects I worked on, but instead with the line: sound_effect = sound_effect / max(abs(sound_effect)); After doing this, it became more straightforward to fine tune our parameters for clearer results.

The next challenge I encountered was in the selection of which effects to implement. When I came across the echo effect, I noticed that the affected sample sounded nearly identical to the original sample after the supposed effect was applied. No matter how much I increased the delay time or mix, I noticed no change. In this process, I began to realize that the issue was inherent to the short, pure toned sample. Because of the short duration, the echo incurred no apparent change. To solve this issue, I simply moved onto another effect: ring modulation. This was immediately clearer in the sample, which led me to pursue it further. Looking back, this could have worked with the original sample instead of the pure tone as we began with, but by the time we changed from the pure tone to the original, we were already satisfied with the effects we implemented.

Ultimately, I was faced with the challenge of nearly all user interactive programs: making user input more fool proof. Because we cannot assume that the user will know how to navigate

through the program, it was important that we added redirection, instructions, and displayed messages throughout the program. Above each input line, we provided dropdown menus with options to choose from: whether it was to enter the harmonies or effect, choose from different harmonies and effects, or exit and hear the sound played. This was also intended to allow the user to manipulate one sound sample without having to rerecord. We found that this was tedious during the testing process, so the implementation of the saved sample also aided us while debugging our effects. With forward and backward navigation, our program allows the user to make mistakes, and continue to manipulate their original sound as if nothing happened.

## 4. Challenges & Solutions: Rhiannon

A few challenges that I faced during this project are as follows. The first was integrating the YIN estimator into our script. The pitch tracking algorithm takes the recorded clip as input and produces the samples per second of said clip. Due to technological errors and the clipping associated with the starting and stopping of a recording through the audiorecorder() function in MATLAB, the first few trials with YIN were inaccurate. After graphing the recorded sound, it was evident that the recording was catching the computer clicks at the beginning before plateauing into the frequency of the note being sung. To fix this, I cut the first few milliseconds of the recording/skipped the first few samples, and also smoothed out the recording to be able to reject impossible pitches that may arise from other discontinuities in the recording.

A second problem that I faced came from our original approach of using the pure tone that we used to layer harmonies, for the implementation of the sound effects. I was hoping to implement reverb, but this is virtually impossible with a pure tone, as it lacks harmonic structure, has no complex, roomy effect, and no sense of space based on how the waves interact. Early and late reflections do not exist, and the sound is not three-dimensional. I decided to instead implement sound effects that could be heard when applied to the pure tone, such as vibrato or a phaser. We changed our implementation to use the original sound recording rather than the pure tone for the sound effect execution, because we realized the pure tone was not necessary to use for ease, considering the sound effects can just as easily be implemented with the original recording. As future work for this project, without time constraints, I would re-implement reverberation, as with the original sound recording, it would be audible.

A third challenge that I faced was the inaccuracy of user input. Firstly, a user may not sing the entirety of the recording, so the skipping of samples/cutting the first few milliseconds of the recording assisted with that. Secondly, a user will likely not sing a note to its exact frequency level. For example, I conducted a test by singing a C. A middle C has the frequency 261.63 Hz, but on my attempt, the frequency was 262.35 Hz. To account for this, I rounded to the nearest MIDI note to be able to accurately tell the user what a triad would be with their note. A possible drawback of this may be if a user were trying to sing a note that was sharp or flat, the program may round to the natural note. This could be addressed as a future improvement.

## 5. Improvements & Future Work (Written by: Rhiannon)

Without time and resource constraints, there are a few improvements we would make to our project. The first would be synthesizing the actual voice recording for the harmonizer, rather than using the pure tone. This was helpful for our implementation with the time we had, but ideally, we would like this idea to work so the user could hear a harmonized version of their voice. A second would be additional sound effects, such as reverb. We used a sample size of sound effects, and we would hope to implement more to make our project more comprehensive. A third improvement would be to change the use of MIDI note numbers, as this is a Western-centric way of looking at musical notes, being based on the 12-tone scale. An enhancement could be to add the possibility of choosing a certain scale or tuning. A fourth change would be to explore the MATLAB GUI, to make our program even more interactive and user-friendly. A possible fifth improvement, which, when discussing project ideas, was one of our original hopes, would be to create the harmonizer in real time. Hoping to attempt this, we learned MATLAB would not be the best platform for this kind of implementation. PlugData, for example, may be better, but we both felt more comfortable using MATLAB. We would want to work to allow the user to add harmonies on top of their voice in real-time/live. We hope to be able to come back to this project in the future and improve it further.

## 6. Workload Breakdown (Written by: Finn)

During the initial phases of the project, we worked closely together on brainstorming, planning, and solidifying our idea. There were several aspects to consider and hash out together that eventually led to the current version of our project. As outlined earlier, we thought about real-time manipulation, using a pure tone versus original sample, and different pitch detection methods to name a few. However, we also clearly delegated work between the two of us with a concrete plan from the get go:

Finn focused on the recording of sound, user interface in MATLAB, layering of harmonies, ring modulation effect, and chorus effect.

Rhiannon focused on pitch tracking, conversion to pure tones, triads / harmony formulas, as well as the vibrato, distortion, and phaser effects.

Throughout the duration of the project, we were also collaborating to debug and get a second opinion on more subjective matters, such as the sound of each affected sample. While we had distinct focuses within this project, we also were able to help each other on a day-to-day basis when needed. This has been an extremely collaborative, yet individually focused project that has allowed us to tap into what we have learned in the class, research further, and apply the combined methods into something useful for others.

## 7. Citations and Contributions

Das, Orchisama, Pitch-Tracking, (2016, 2023), GitHub repository, yin_estimator.m

https://github.com/orchidas/Pitch-Tracking/blob/master/yin_estimator.m

de Cheveigné, A., & Kawahara, H. (2002). YIN, a fundamental frequency estimator for speech and music. The Journal of the Acoustical Society of America, 111(4), 1917–1930. https://doi.org/10.1121/1.1458024