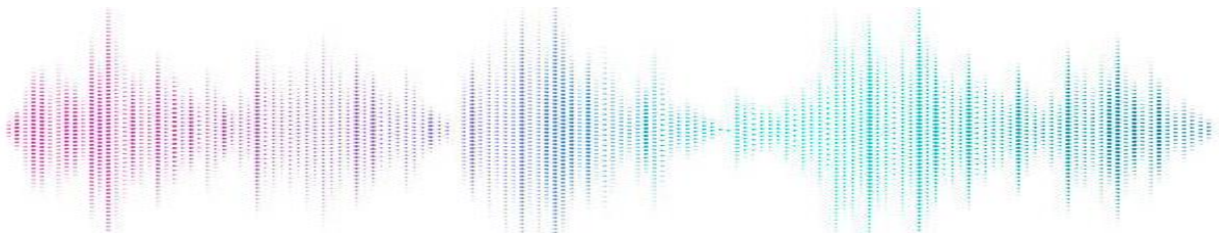
VOICE RECORDING AND ANALYSIS



DIGITAL SIGNAL & PROCESSING

FINAL PROJECT REPORT

GROUP 17

Nguyễn Thị Khánh Huyền	BA10-025
Vi Văn Hiếu	BA10-019
Vũ Yến Linh	BI11-152
Đoàn Thuận An	BI11-002
Lê Duy Long	BA10-033
Trần Xuân Đại	BI7-029

I.Introduction:

1. Project's topic:

Voice recording and analysis.

2. What is special about this program?

The special feature of this program is that it bases on changing and interacting with the frequency of the signal to analyze the input sound and create a better output with the expected effects.

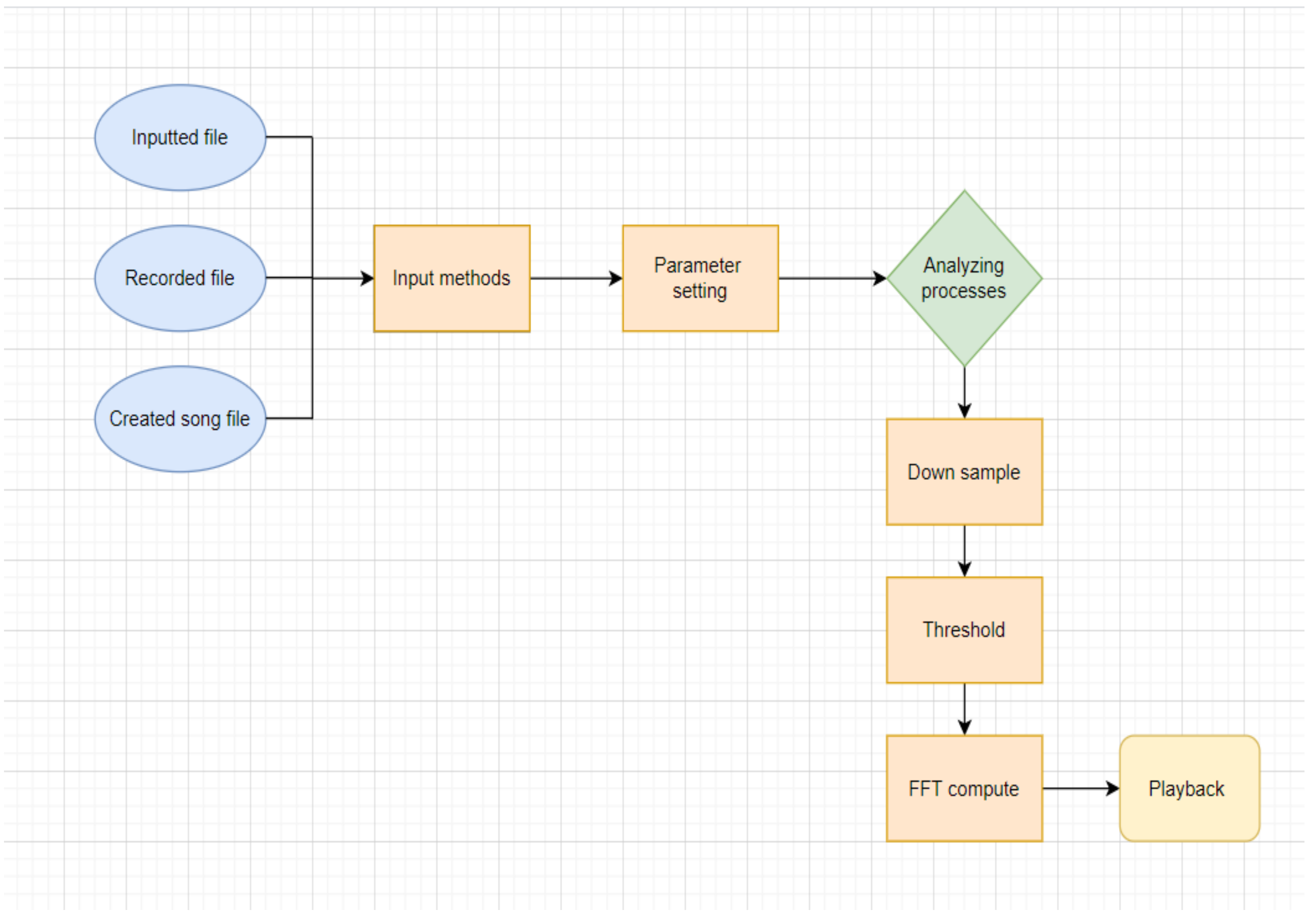
3. What can this program do?

This is a multifunctional interactive program that oriented to human voices in particular and audio files in general. Its possible operations include:

- Record audio directly from the microphone and saving it to the device.
- Create a song or melody.
- For directly recorded or imported audio files:
    - File analysis: Read files, analyze signals, plot graphs, reduce audio frequencies, filter noise, calculate frequencies.

4. An overview:

To sum up, our program gives the user 3 input methods and allows them to analyze the audio files through the functions we have developed as well as to achieve an output the user wants. The flow chart below will give you an avowed overview of this program



II. How does this project work?

It can be explained in each of the divided sections, step by step.

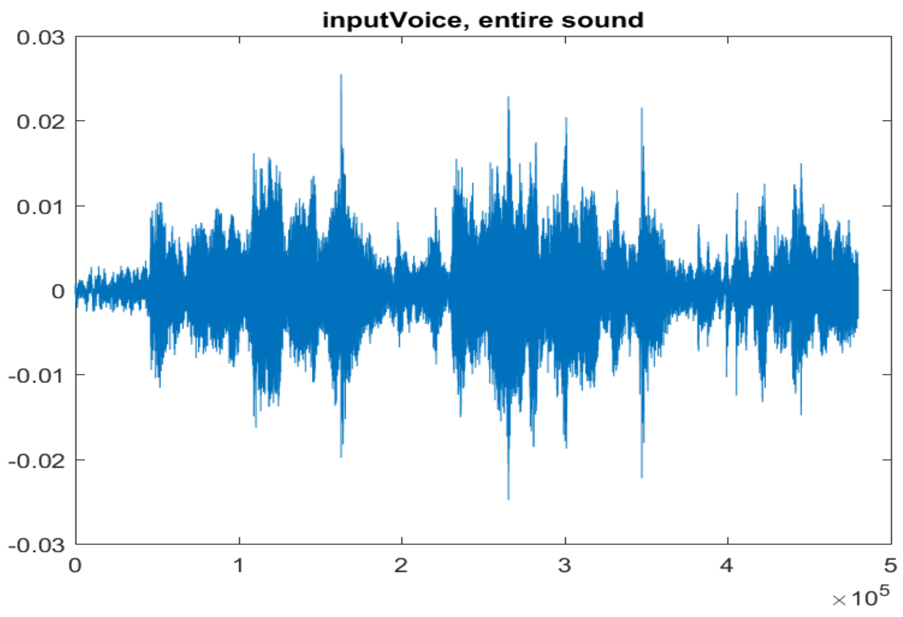
1. Input methods:

1.1 Input file:

To input a recorded audio file, first, we need to have the file in the current Matlab folder. Use the audioread command to read the inputted file in ‘wav’ format. We can adjust the speed by altering the frequency, in other words, changing its fs.

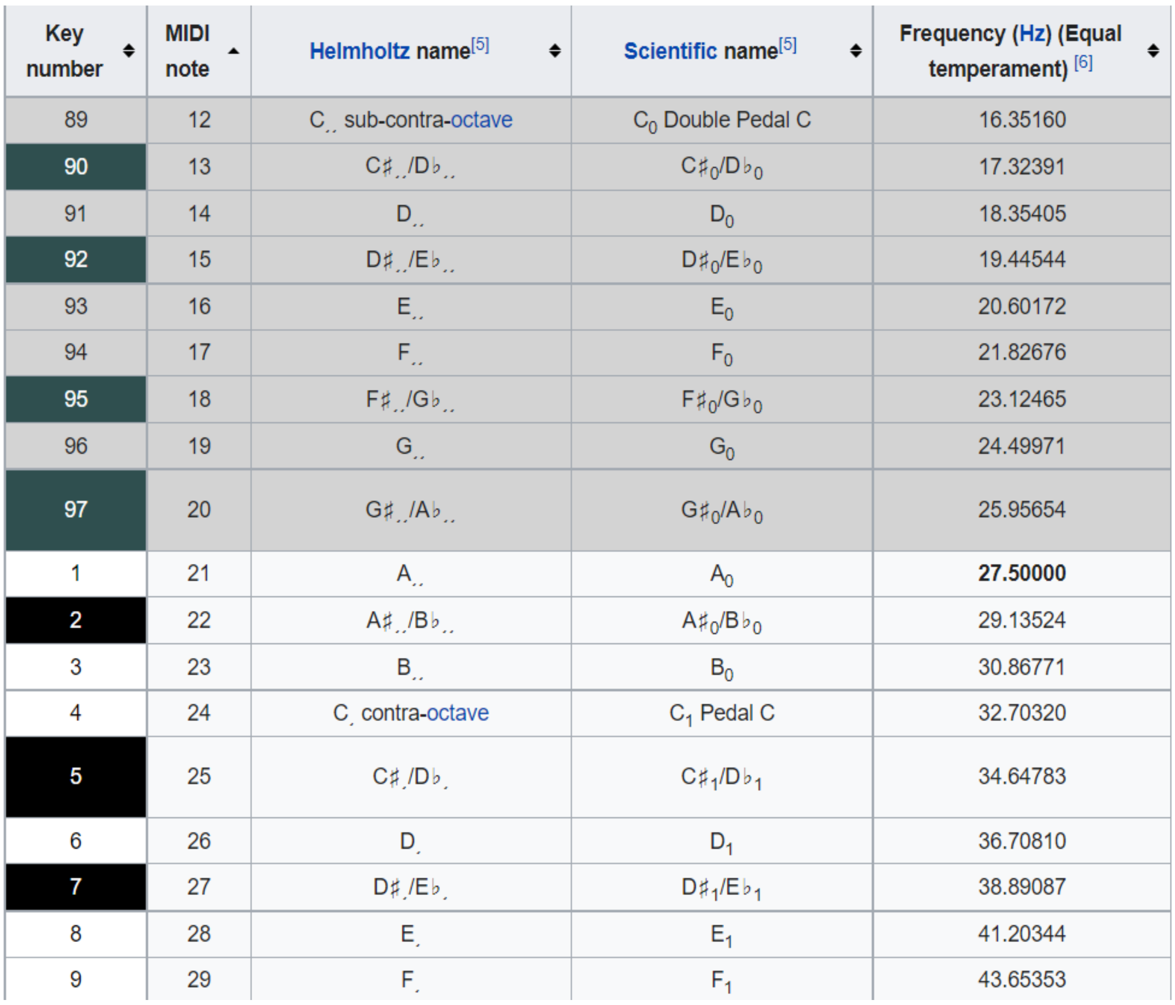
1.2 Voice recording

To record a voice directly from the microphone, we used the function audiorecorder. First of all, we specified the sample's frequency(fs), the number of channels equal to 2(channel) which returns stereotype, and finally the numbers of bits (bits) to store the audio file for 10 seconds (duration). Then we passed the first 3 elements into the function and the record will be blocked using recordblocking with the duration value passed in. The number of bits was used to store audiofile, we can adjust it to 8, 16, or 24. Finally, it was using the function getaudiodata to store the recorded audio signal in a numeric array and write the file in wav format using audiowrite.



1.3 Create a song file:

Using function “wave = key(p, n, fs)” with p is the key number index (idx = 440*2^((p-49)/12)), n is the duration, we can create a simple pitch using the key number of the piano key's frequencies. We tried the song Happy Birthday.



Key number	MIDI note	Helmholtz name[5]	Scientific name[5]	Frequency (Hz) (Equal temperament) [6]
89	12	C͵͵ sub-contra-octave	C_0 Double Pedal C	16.35160
90	13	C♯͵͵/D♭͵͵	$C♯_0/D♭_0$	17.32391
91	14	D͵͵	D_0	18.35405
92	15	D♯͵͵/E♭͵͵	$D♯_0/E♭_0$	19.44544
93	16	E͵͵	E_0	20.60172
94	17	F͵͵	F_0	21.82676
95	18	F♯͵͵/G♭͵͵	$F♯_0/G♭_0$	23.12465
96	19	G͵͵	G_0	24.49971
97	20	G♯͵͵/A♭͵͵	$G♯_0/A♭_0$	25.95654
1	21	A͵͵	A_0	**27.50000**
2	22	A♯͵͵/B♭͵͵	$A♯_0/B♭_0$	29.13524
3	23	B͵͵	B_0	30.86771
4	24	C͵ contra-octave	C_1 Pedal C	32.70320
5	25	C♯͵/D♭͵	$C♯_1/D♭_1$	34.64783
6	26	D͵	D_1	36.70810
7	27	D♯͵/E♭͵	$D♯_1/E♭_1$	38.89087
8	28	E͵	E_1	41.20344
9	29	F͵	F_1	43.65353

Firstly, we set the rests respectively by 1.0, 0.5 and 0.25 second, which is the duration of one’s pitch.

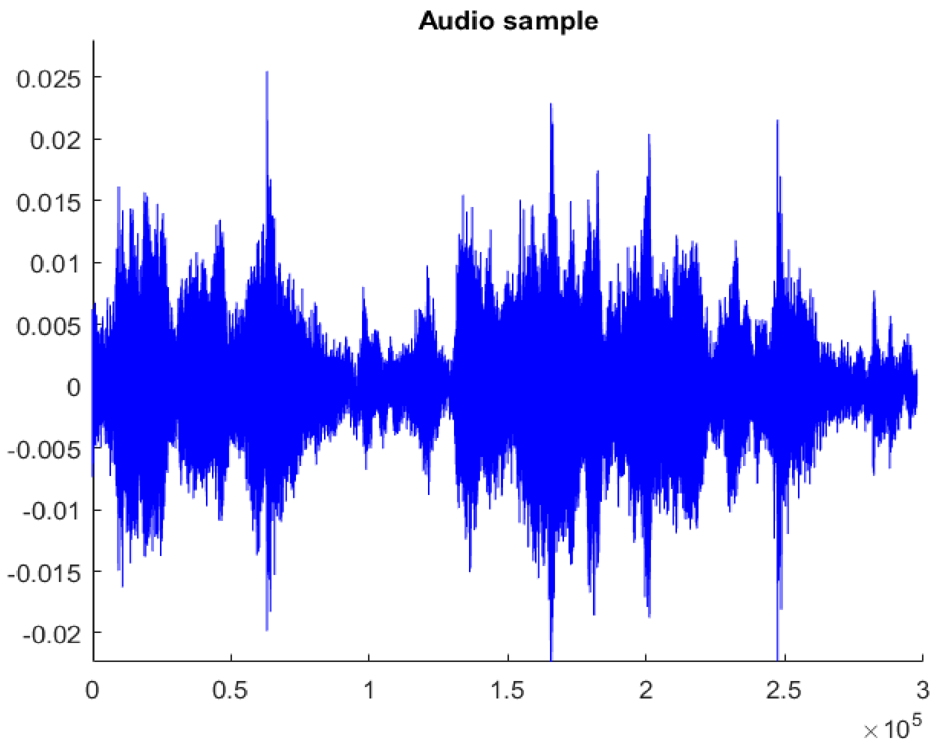
By then, we created two pitches, the first one with 0.5s duration and the second is 1.0s. Using the wave function to store the key’s frequencies into the respective notes.

At long last, we stored the notes in an accurate orders into the phases to create a music sheet and do the same with the phases to create the final song file.

2. Analyzing audio file:

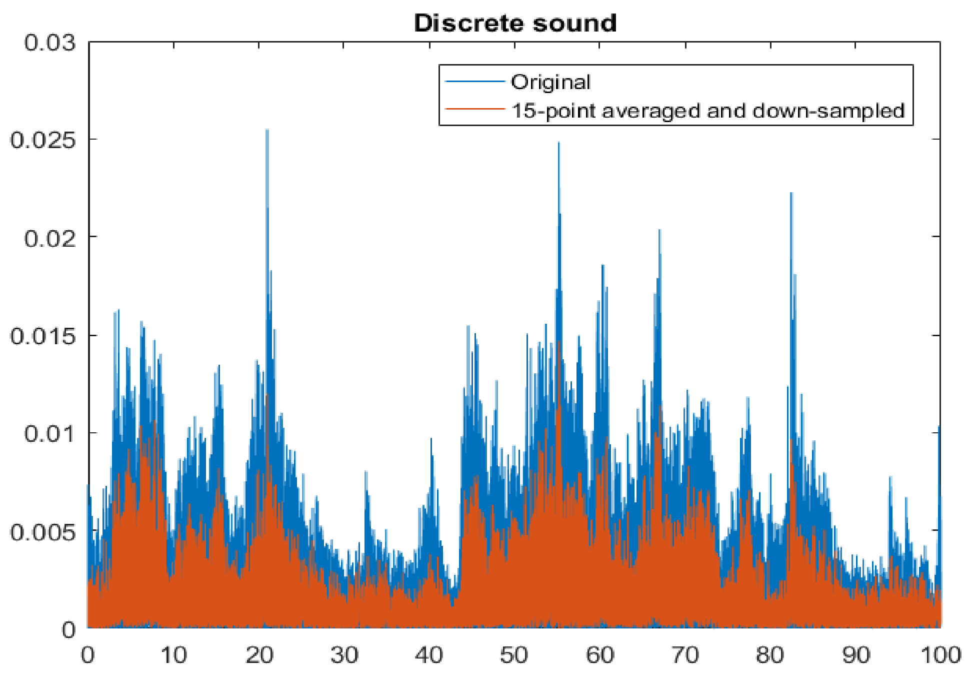
2.1 Parameter setting:

In this section, we used the linspace function to control the number of points, including the endpoint of the sample by setting the startpoint as t1 and the endpoint as t2 respectively, step n. Then, we use audiowrite command to first hear and check out the original sample.



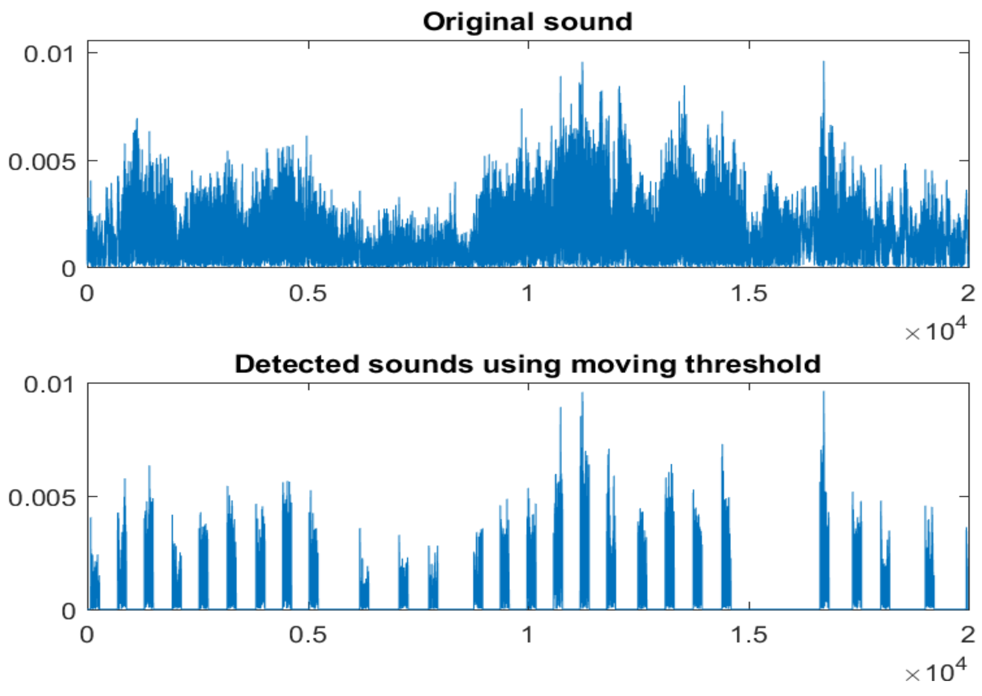
2.2 Down Sample:

Firstly, use an average filter to remove some of the high-frequency sound points and decrease the number of points in the sample to speed up the next-step FFTs calculations**.**



2.3 Threshold the sound points and compute the frequency:

Since some of the sound points in the recorded file are vitally higher than the others. We used a moving average threshold to detect these high points while still preserving some of the smaller but not less important. Went through all points and computed the FFT. This will compute every sound points in the audio track that we have cut in step 2.1.



2.4 Playback the sound points:

The last section is to go through all the steps above and return a modified sample without losing much quality compared to the original sample.