EF513 : Introduction to Music
Mini Project : Spectrum analysis

Gihan Jayatilaka* E/14/158

25/05/2020

1 Implementation :Write a computer program (in Matlab or any other language) to extract the fundamental frequency and the harmonic partials of wave forms that belong to the following instruments Guitar, Flute, Violin.

Note: The absence of references for this section is because the algorithm was written from first principles.

1.1 Algorithm

Data: audio_file
Result: $\mathrm{f}_0, f_n[], A_n[]$
initialization;
$[\mathrm{audio}, \mathrm{F}_s] \leftarrow read(audio\_file)$
$[\mathrm{A}_n.real, A_n.img] \leftarrow fast\_fourier\_transform(audio)$
$\mathrm{A}_n \leftarrow \sqrt{A_n.real^2 + A_n.img^2}$
$\mathrm{f}_n \leftarrow calc\_freq(F_s)$
$[\mathrm{A}_n, f_n] \leftarrow positive\_freq\_only(A_n, f_n)$
$[\mathrm{A}_n, f_n] \leftarrow find\_probable\_partials(A_n, f_n)$
$[\mathrm{A}_n, f_n] \leftarrow refine(A_n, f_n)$

Algorithm 1: Pseudocode for the full algorithm

*gihanjayatilaka@eng.pdn.ac.lk

Data: $A_n[], f_n[]$
Result: (A,f) pairs for probable partials
initialization;
prorbable_partials ← $\{emptyset\}$
for $i = 0;\ i < 10;\ i = i + 1$ **do**
    peak ← $find\_peak(A_n)$
    probable_partials.append(($A_n[peak], f_n[peak]$))
    $A_n[small_window_near_peak] \leftarrow 0$ **end**

Algorithm 2: Pseudocode for find_probable_partials

Data: probable_partials
Result: refined list of (A,f) pairs for probable partials
initialization;
refined_list ← $\{emptyset\}$
for $f \in probable\_pairs$ **do**
    Assume f is the fundamental frequency;
    error_f ←$\sum$ inharmonic_parial_amplitude × deviation_from_partial given f;
end
$f_0 \leftarrow min\_error\_f$
probable_partial_freqs ← $find\_freqs\_close\_to\_integer\_multiples\_of\_f_0$
probable_partial_freqs ← $filter\_by\_amplitude\_thershold(probable\_partial\_freqs)$

Algorithm 3: Pseudocode for refining probable_partials

1.2 Implementation

The solution is implemented in python using the following tools.

Programming language	Python 3
Audio file reading	librosa (that uses ffmpeg to read mp3)
Plotting	Matplotlin [1]
Numerical operations	Numpy [2]

1.3 Files included

analyze.py	This is the python program
run.sh	The shell script that runs everything
firstrun.sh	The shell script installing the required packages for python
guitar.wav, violin.wav, flute.mp3	audio input files
instru/timeSeries.png	Time series signal
instru/freqSpectrum.png	Fourier spectrum
instru/partial-finding-XX.png	Iterations of algo 2
instru/harmonics.png	Output of algo 3
instru/output.txt	Fundamental and harmonic freqs with amplitudes
instru/log.txt	Log file for debugging

1.4 Results

1.4.1 Fundamental frequencies

Instrument	Fundamental freq
Violin	246.11 Hz
Flute	521.36 Hz
Guitar	184.41 Hz

1.4.2 Harmonic partials

	n	fn	An
✓	1	521.36	12745.74
✓	2	1043.38	1826.32
✓	3	1564.75	2357.27
	4	2086.33	89.16
	5	2597.90	126.14
	6	3129.71	298.56

Table 1: Flute

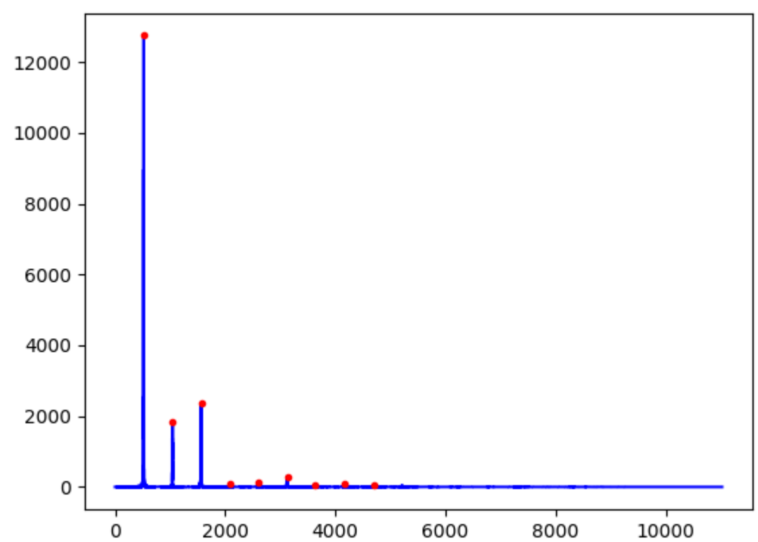

Figure 1: Harmonics of the flute

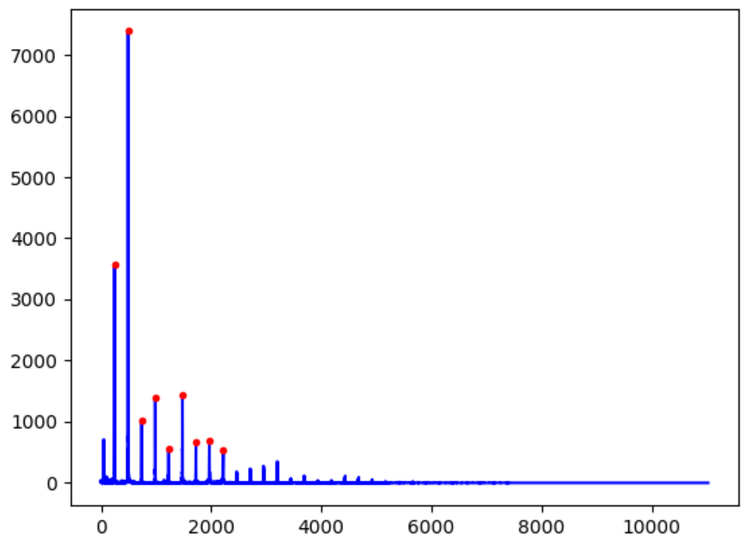

Figure 2: Harmonics of the violin

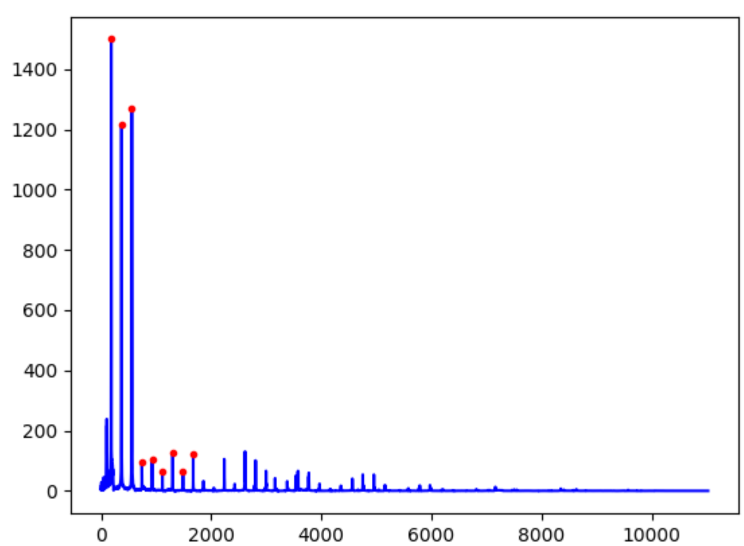

Figure 3: Harmonics of the guitar

	n	fn	An
✓	1	246.11	3561.42
✓	2	492.23	7388.57
✓	3	737.96	1010.52
✓	4	984.83	1404.65
✓	5	1230.94	559.03
✓	6	1477.80	1447.17
✓	7	1723.92	658.62
✓	8	1967.78	685.11
✓	9	2216.52	532.85

Table 2: Violin

	n	fn	An
✓	1	184.41	1499.67
✓	2	371.83	1215.52
✓	3	557.24	1269.36
	4	741.65	93.65
	5	928.06	105.03
	6	1113.47	65.64
	7	1300.89	127.71
	8	1486.30	65.40
	9	1673.72	123.68

Table 3: Guitar

1.5 Insights

1.5.1 The sound file is not 100% uniform throughout the time period. Can we crop the sound file for better accuracy?

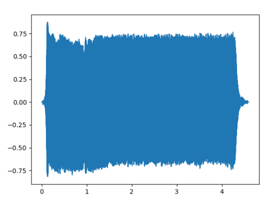

Figure 4: Flute time series

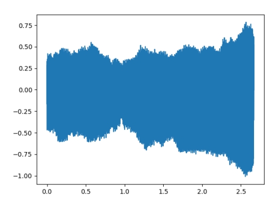

Figure 5: Violin time series

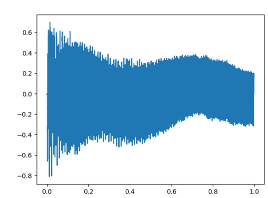

Figure 6: Guitar time series

Time series images (4,5,6) shows that violin and guitar are not giving the same pattern through out the file. But manually cropping this prevents the algorithm from being automatic. The refining part of the algorithm is used to handle this issue.

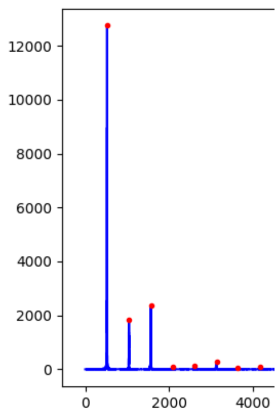

Figure 7: Flute harmonics

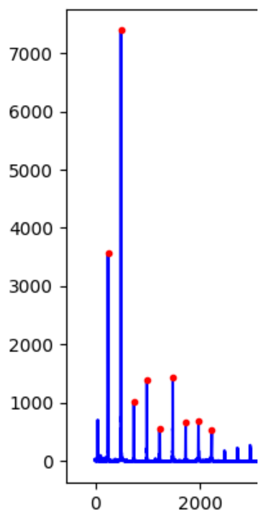

Figure 8: Violin harmonics

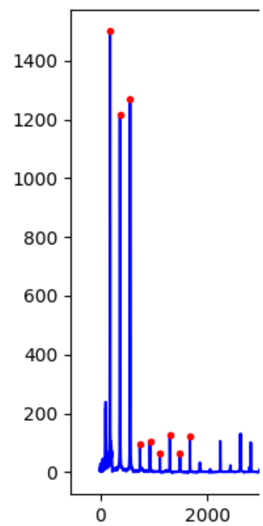

Figure 9: Guitar harmonics

This figures (7,8,9) shows how the first peak of the Flute is considered as the fundamental frequency (marked by a red dot) while the first peak of other two instruments are discarded by the algorithm. This is done by **ALGORITHM 3** without being explicitly programmed to look for the noise only in this region. This solution is more generalized.

1.5.2 The harmonics might have slight shifts from the expected frequencies. How are they handled?

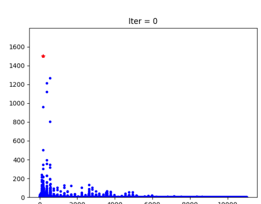

Figure 10: Algorithm 2 iteration 0

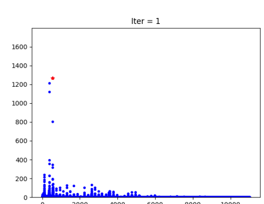

Figure 11: Algorithm 2 iteration 1

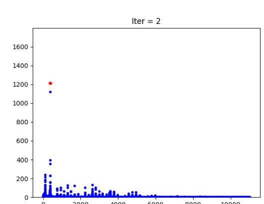

Figure 12: Algorithm 2 iteration 2

The peaks in the frequency spectrum are not impulse like functions. They are spread out in a small window of frequency. This issue is handled in **ALGORITHM 2** by clearing out the neighbourhood of the peak before finding the next one. This is shown in figures 12, 11 and 12

2 Written question

2.1 What is the importance of Fourier transform with relation to digital music (200 words)

Fourier transform is a mathematical tool capable of transforming a signal in the time series (music in the context of this project) to its counterpart in the frequency domain. The transform itself will generaly project any such signal and might give a dense frequency spectum throughout the frequency domain. These spectrums are difficult to analyze with infinite accuracy due tot he obvious limitations of computing (while approximations are possible). But when the time series signal is periodic (which is characterized by a fundamental period T_0) the corresponding frequency spectrum becomes very sparse (having zeros throuhout the spectrum except for a few specific frequencies known as harmonics).[3]

Music is generally periodic signals (at least periodic for a small time period). The fourier transform of of music signals is informative (we can easily deduce what note is being played), interpretable (since most components are zero, easy to understand) and manipulatable (eg: we can do filtering easily). When it comes to digital music (which is music sampled at discrete points in time at discrete representable floating point numbers) the fourier transform could be used for common purposes like compression, noise removal, enhancement, synthesis and transmission. Some uncommon usecases of the transform could be source separation, instrument identification, genre identification and vocal quality evalation.

2.2 What is timbre?

The basic way of characterizing the music (sound) is by it's frequency (fundamental – smallest among harmonics or dominant – highest amplitude) and amplitude. But these measures can be the same for any instrument playing the same note at same frequency. Obviously, human ear can find a difference. Timbre is an attempt to characterize this difference. [4],[5]

Timbre is also known as the tone quality. This is mathematically characterized by the frequency components of the sound in all the harmonics. For example, two instruments can have the same dominant frequency but very different amplitudes at other overtones. In a way, if it was not for timbre, all musical instruments will be equivalent to tuning forks.

2.3 What is the fundamental frequency of a waveform?

Fundamental frequency is the smallest frequency among the harmonics (other harmonics being overtones, who are integer multiples of the fundamental frequency). Fundamental frequency is also the time period of the periodic signal (in digital music, the signal is considered to be periodic at least over a small period of time.)

2.4 What do you understand by harmonic partials and in-harmonic partials?

Consider that there is a time series $f_{(t)}$ which can be written as a summation of sinusoidals as

$$f_{(t)} = \sum_{n=0}^{n=N} A_n sin(2\pi f_n t);\ \text{where } A_n \neq 0$$

. These f_n s can take any value.

Assume that the function is periodic with a time period T_0 such that

$$f_{(t)} = f_{(t+kT_0)};\ \text{for } k \in \mathbb{Z}$$

The fundamental frequency f_0 would be $f_0 = \frac{2\pi}{T_0}$. The positive integer multiples of f_0 can be written as $\{kf_0; k \in \mathbb{Z}^+\}$

Harmonic partials	$A_n sin(2\pi f_n t)$ for $f_n \in \{kf_0; k \in \mathbb{Z}^+\}$
In harmonic partials	$A_n sin(2\pi f_n t)$ for $f_n \notin \{kf_0; k \in \mathbb{Z}^+\}$

2.5 What is the relationship of harmonics partials and timbre?

If there is only one harmonic partial – dominant frequency ($N = 1$) the instrument will be playing the pure note. But the existence of a large number of harmonic partials ($N >> 1$) gives rise to timbre.

References

[1] J. D. Hunter. Matplotlib: A 2d graphics environment. *Computing in Science & Engineering*, 9(3):90–95, 2007.

[2] Pauli Virtanen, Ralf Gommers, Travis E. Oliphant, Matt Haberland, Tyler Reddy, David Cournapeau, Evgeni Burovski, Pearu Peterson, Warren Weckesser, Jonathan Bright, Stéfan J. van der Walt, Matthew Brett, Joshua Wilson, K. Jarrod Millman, Nikolay Mayorov, Andrew R. J. Nelson, Eric Jones, Robert Kern, Eric Larson, CJ Carey, İlhan Polat, Yu Feng, Eric W. Moore, Jake Vand erPlas, Denis Laxalde, Josef Perktold, Robert Cimrman, Ian Henriksen, E. A. Quintero, Charles R Harris, Anne M. Archibald, Antônio H. Ribeiro, Fabian Pedregosa, Paul van Mulbregt, and SciPy 1. 0 Contributors. SciPy 1.0: Fundamental Algorithms for Scientific Computing in Python. *Nature Methods*, 17:261–272, 2020.

[3] Dr. Hugo Reinman. *Dictionary of music.* Augener, London, 1908.

[4] Richard Lyon Shihab Shamma. Auditory representation of timbre and pitch. *Auditory Computation. Springer.*, In Harold L. Hawkins Teresa A. McMullen (eds.).(10):221–23, 1996.

[5] Harold S. Powers. *Melody, The Harvard Dictionary of Music, year =*.