Time Domain Pitch Recognition

Michael G. Chourdakis, Haralampos C. Spyridis
Music Department, University of Athens, Greece.
{m@turboirc.com, hspyridis}@music.uoa.gr

***Abstract* — The power of Time Domain based methods over Frequency Domain methods for signal processing and pitch recognition; The suggestion of new Notation Symbols in order to represent all possible frequencies in the European, Byzantine, or Electronic (MIDI) notation.**

I. Introduction

The following will be described:

1. The advantages of time-domain methods over the frequency-domain methods in digital signal processing.
2. The methods used to identify the pitch, and the collection of the results to either a frequency vector, or to music notation.
3. The success probabilities of the methods, and the problems that may be encountered at the process.
4. The creation of new music notation symbols, needed to describe music notes that they have slightly different frequencies than the standard European frequency vector; these symbols will equally apply to other notation systems, like the Byzantine Notation or the e-notation (MIDI).
5. The application of the methods by experimenting.

The end result will be the ability to detect and display the pitch of a signal automatically from the PC, without the aid, recommendation, or otherwise interference of a human being, which may be less or more influenced by music knowledge and/or music experience.

Furthermore, we will be able to write down the recognized pitch with new symbols, to any main notation scheme (e.g. The European Notation, the Byzantine Notation, the MIDI notation etc).

There are many applications for our method:

1. We can check and fix an instrument's accurate tuning, either a conventional instrument or an electric one.
2. The ability to verify the correct performance of a musical piece, e.g. the ability to check if a singer will properly sing their song, based on known estimated European, Byzantine or other specific-frequency climaxes.
3. The ability to write down any sort of simple or complex vector of music symbols (notes) with altered frequency, in a compatible format which is easy understood and mastered by the music student.

II. The Sound Signal

The sound signal generated from a sound source (voice, instrument) is always a function of time:

$$S(t) = A_0 \sin(2\pi f_0 t) + A_1 \sin(2\pi f_1 t) + \ldots + A_\nu \sin(2\pi f_\nu t) \quad (1)$$

Where A1, A2, A3 ….are the local peaks of each oscillation (amplitudes), and f1, f2, f3… fν are the frequencies generated by the sound source. In theory, a sound source generates an infinite number of signals with decreasing amplitude; in practice, we are only interested in the few first frequencies because all others' amplitude is practically zero.

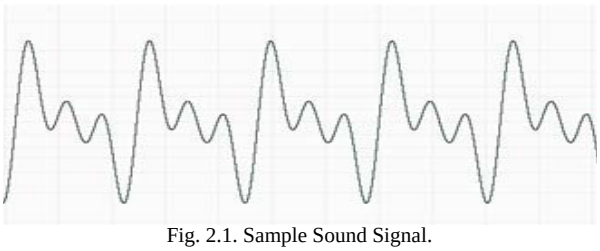

Fig. 2.1. Sample Sound Signal.

A sound signal passing through an ADC (Analog to Digital Converter) and sampled with a Sampling Frequency S, will have, according to the Niquist Theorem, a limited number of possible frequencies not over the value S/2 [1].

That signal would be periodic if each of the generated frequencies was an integer multiply of the first generated frequency, since for the equation (1) it is:

$$S(t) = 0, t \in \Re$$
$$\sin(x) = 0 \Leftrightarrow \sin(\nu x) = 0, \nu \in \Re \quad (2)$$

III. The Fourier Transform

The Continuous Fourier Transform (CFT) of a continuous function of the time f(t) is a function of frequency/intensity, described by the following formulas:

$$f(\omega) = \int_{-\infty}^{+\infty} f(t) e^{-i\omega t} dt \quad (3)$$

$$f(\omega) = \sum_{t=-\infty}^{+\infty} f(t) e^{-i\omega t} \quad (4)$$

Equation (3) is for continuous signals, equation (4) is for discrete signals.

Fig 3.1. Signal represented as a function of time

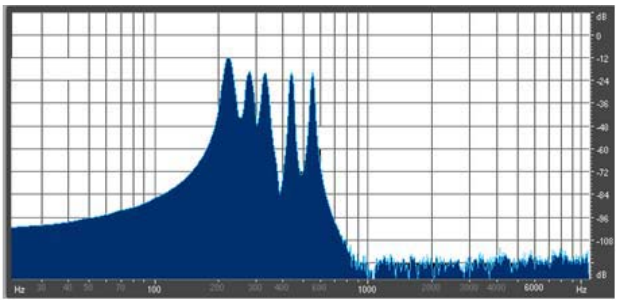

Fig 3.2. Signal represented as a function of frequency

Due to the Heisenberg Uncertainty Principle, we are not able to detect a frequency at a given point of the signal, but we have to apply a transformation Window to the signal in order to perform the Fourier Transform. Each of the available Windows has its advantages and its disadvantages; our main problem is that we cannot detect the frequency with great accuracy.

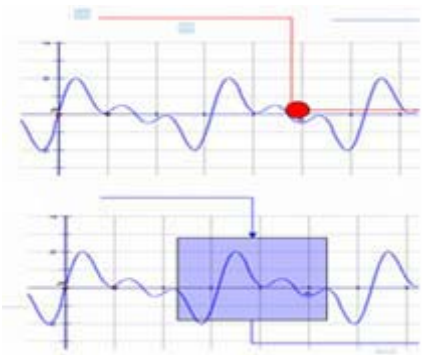

Fig. 3.3. Applying a Window

To use the Fourier Transform and to work on the Frequency Domain would apply the following limits:

1. The Fourier Transform works best with periodic signals. In theory, all signals generated by sound sources should be periodic; in practice, none of them are. The result of this is the 'Gibbs Phenomenon', the 'leaks' generated by the Fourier Transform's inability to work with such signals.

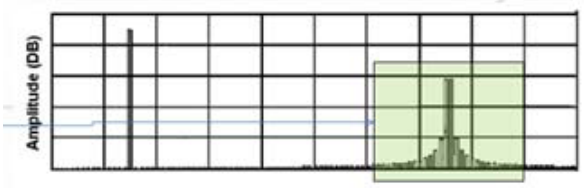

Fig 3.4. Gibbs Phenomenon

The second limitation of the Fourier Transform applies to the discrete version of it. The Discrete Fourier Transform, used in discrete signals and in the PCs, accepts as input a fixed number of samples and returns the same number of complex numbers [2].

The magnitude of each of these complex numbers describes the intensity of a specific frequency found in the signal. This frequency can be estimated by the formula:

$$f_i = \frac{iSR}{N} \quad (5)$$

Where SR is the sample rate of the signal, and i is the index of the frequency in the complex vector returned by the Fourier Transform. So we can immediately see that the Fourier Transform can only return information for a fixed number of frequencies (Between 0 and SR/2) each of them must be calculated from the above formula for integer values of i.

What happens to the frequencies found for non-integer values of i? Their magnitudes are mixed with the closer frequency, which would result in false reports by the Fourier transform, especially when trying to detect very slight pitch changes [3].

All these would force us not to use Frequency Domain methods for accurate pitch estimation, but rather use Time Domain methods instead.

The main advantages of Time Domain methods are:

1. They can give better results for non-periodic signals.
2. They can work within a limited collection of samples.
3. Their results obtained are accurate, and accuracy increases with higher sampling rate values.

The main difficulty in using the Time Domain Methods is speed. Signals have to be filtered by less or more complex filters in order to be analyzed in the Time Domain Mode; this filtering and preprocessing, as also the Time Domain Processing itself can take a lot of time.

Fortunately, with the high-power modern PCs at our service nowadays, we need not anymore worry about the time performance of our methods.

IV. Pitch Recognition

We record the signal from the microphone or other sound input. After recording, the signal must be pre-processed.

Preprocessing involves either low pass filters to cut off unneeded frequencies (If we can assume that there are unneeded frequencies, for example, when recording a human voice which has a limited frequency bandwidth) , or other algorithms, more or less complex, aiming to simplify the signal as much as possible [4].

After that, the signal is analyzed by our main algorithm which defines a structure, describing the frequency, the time and the amplitude, and generates a vector of items describing those elements for the whole signal.

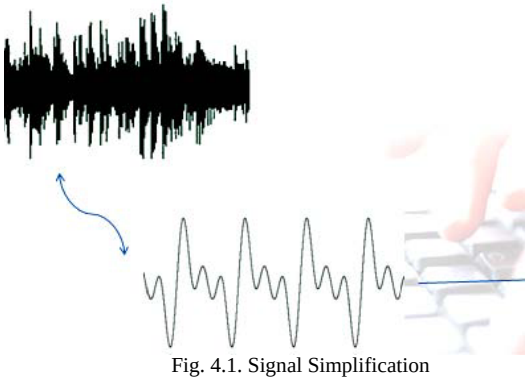

Fig. 4.1. Signal Simplification

The next step is to analyze, connect, possibly edit, and possibly reject those results, in order to convert the full signal to the music notation. This music notation can be either the European:

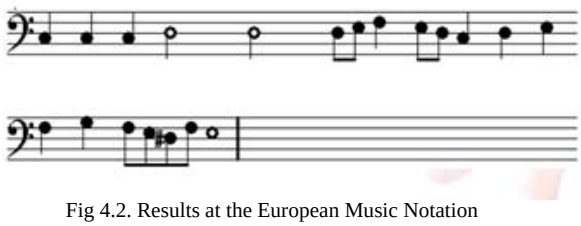

Fig 4.2. Results at the European Music Notation

Or, it can be the Byzantine Notation, which is a notation describing relevant steps between notes:

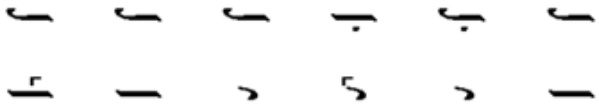

Fig 4.3. Byzantine Notation

V. The Problems

In theory, every signal should be clean and easy to be analyzed. In practice, almost each of the recorded signals has its problems.

The most common problems encountered are:

The signal's amplitude is very low; In which case it must be amplified in order to be further analyzed:

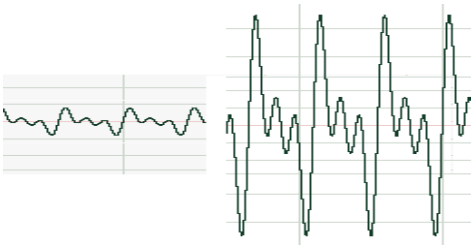

Fig 5.1. Amplifying the signal.

a) One of the frequencies of the signal has greater intensity than the fundamental frequency f0. This problem, one of the reasons that Fourier Transforms also fail, is resolved by analyzing the signal by convolution and statistical methods to detect if there is a case of a fundamental frequency with a lowered intensity.

Fig 5.2. The Fourier transform shows the first (fundamental) frequency f0 weaker than f1.

b) The signal has so much noise that it cannot be analyzed further without noise reduction. This usually involves high-pass filtering the signal in order to remove as much noise as possible, before retrying the recognition process.

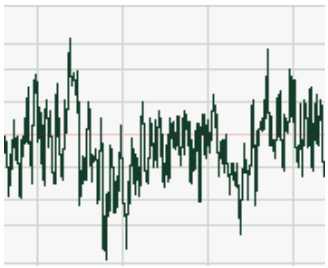

Fig 5.3. Noisy signal

c) The signal might be unstable. For example, the human voice is almost always unstable, i.e. when attempting to sing the note 'C', the human voice doesn't keep a fixed frequency. In that case our methods involve statistical analysis and the 'amplitude' hint to detect if there is an actual change to the frequency, or it might be a case for a human instability feature like vibrato [5].

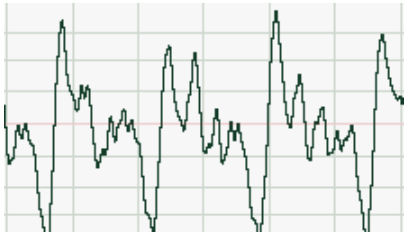

Fig 5.4. Unstable signal

VI. Current Notation Problems

At the current European notation scheme, there is no way to represent a note of which its frequency is slightly altered.

For example, we can try to represent the Frequency 130.81 Hz using the note 'C', and the frequency 138.59 Hz using the note 'C#'. But, we are not able to draw a symbol that would represent any frequency between these two numbers:

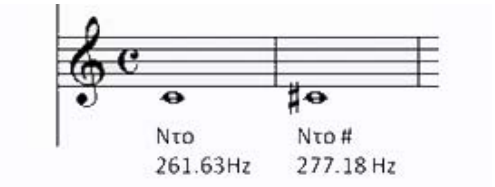

Fig 6.1. Frequencies of European notes.

When taking MIDI into consideration, we have the very same problem, since in MIDI, the 128 available notes are mapped to the European notes (From A0 to B7).

The same problem occurs to the Byzantine Notation. The Byzantine notation is a relative representation of the symbols; each symbol is not a fixed note, but rather an indication of how many notes we should go up or down. For example, When we are in 'C 130.81' (Byzantine 'NH'), and we encounter the 1 (Byzantine Symbol 'OLIGON'), we should get to the next note, in which case it is the Byzantine 'PA' (D). The frequency of that D

is not fixed (as in the European notation), but it depends on the mode definition; in byzantine music, there are 8 different modes and the octave is divided into 72 equal parts (and not 12 semitones, as in the European). In the following example, PA is 146.83. But we still have the problem to represent, say, 144Hz.

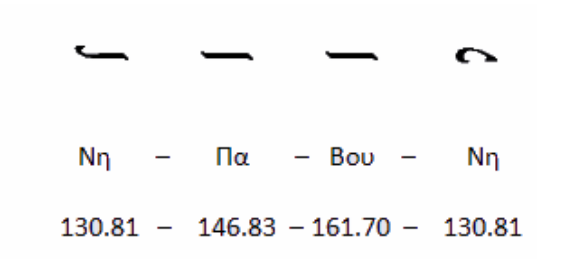

Fig 6.2. Byzantine notes pronounced 'NI', 'PA', 'VOU' and their frequencies for a specific Byzantine mode.

VII. The New Notation Suggestions

A proper solution to the above problems should meet the following restrictions:

1. It should be easily to understand and learn.
2. It should be subject to further revision.
3. It should be possible to be ignored by human beings or instruments not designed to take advantage of it, and/or take advantage of the micro-tuning.
4. It should be compatible with existing symbols and notation.
5. It should not be confused with existing symbols of sharp and flat.
6. It should be able to divide the entire octave to any defined number of parts, up to the international standard, the 'cent'.
7. It should take as little space as possible, in order not to confuse the reader.
8. It should be usable in PCs and existing sound and score processing applications.

We define 2 new symbols: The *Simple Sharp '/'* and *the Simple Flat '\'*.

We define, as a new symbol for a single sharp, the symbol of '/' (forward slash), to be placed to the left of a note:

Fig 7.1. Simple Sharp

More slashes at the left of the note would suggest double, triple sharp etc.

We define, as a new symbol for a single flat, the symbol of '\' (Backward slash) to be placed to the left of a note:

Fig 7.2. Simple Flat.

More slashes would suggest double, triple flat etc.

In the rare case we might need more sharps/flats than the available space permits; we can use a number before the simple sharp/simple flat symbol.

The simple sharp/simple flat symbols can co-exist and can be combined with the existing traditional sharp/flat symbols:

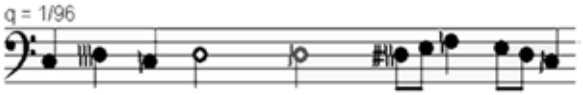

Fig 7.3. Examples of the new symbols.

We define the symbol "q" to indicate the number of sections that the octave should be divided. A musical score can contain an indication with the "q" symbol at the start, or at any other place. The syntax is:

$$q = a/b$$

Where b = the number of sections that the octave should be divided, and a = the number of sections assigned to a simple sharp or simple flat.

For example, q = 1/24 would indicate that the octave is divided to 24 sections, and a single sharp or flat should add/subtract 1/24 to the frequency of the note they were applied.

In case that there is no q indication in the music score, the default is q = 1/12, in which case, the simple flat and sharp are equals to the traditional flat ♭ and sharp #.

The same symbols apply to the Byzantine notation. We put them *above* the byzantine symbols:

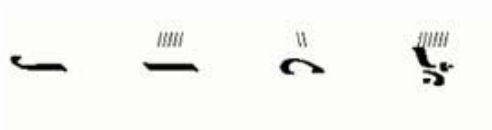

Fig 7.4. Byzantine symbols with simple sharps/flats.

The same symbols are applied to MIDI. We define:

a) A recommendation at the start of each MIDI track (which may reoccur anywhere at the track), to indicate the division, using a META-Event (Midi 0xFF message).

b) A controller message (MIDI 0xBX message) to be placed before each MIDI note message (MIDI 0x9X) in order to alter its frequency by a number of simple flats/sharps.

These extensions are compatible with existing MIDI software, because applications that are not able to recognize our new symbols will simply ignore them and play the MIDI file as if those symbols wouldn't exist.

As a result, we have created a simple method of representing any frequency we need to the musical score, which is easy for both human beings and software to interpret.

References

[1] R.W.Hamming, "Numerical Methods for Scientists and Engineers" *Dover Publications*, New York, 1986, pp. 557 – 560.

[2] Francis J. Flanigan, "Complex Variables, Harmonic and Analytic Functions" *Dover Publications*, New York, 1983.

[3] Richard A. Silverman. "Introductory Complex Analysis". *Dover Publications*, New York, 1984, pp. 1 – 24.

[4] R.W.Hamming, “Digital Filters” *Dover Publications*, New York, 1998.

[5] Alan V. Oppenheim, “Discrete-Time Signal Processing”, *Pearson Education Publishing*, 2001.

[6] Steve Smith, “The Scientist and Engineer's guide to Digital Signal Processing” California *Technical Publishing*, 1997.