Digital Music to Sheet Music
Hugh Smith
Computer Systems Lab 2009-2010

Abstract

Electronic creation of music has become a wide-spread hobby or profession. Electronically generated music is when a user inputs data about a sound into a computer, and it produces said sound. Even though this is very widespread, the opposite is not true. That is, the conversion of raw sound data into sheet music, or something that defines the sound, is much less prevalent. Why is this? It is, of course, very challenging to do. This project aims to do just that - convert a piece of digital music (a .WAV file) into actual sheet music, using the Fourier transform algorithm.

ABC Code → Sheet Music

```
T:Paddy O'Rafferty
C:Trad.
M:6/8
K:D
dff cee|def gfe|dff cee|dfe dBA|\
dff cee|def gfe|faf gfe|1 dfe dBA:|2 dfe dcB|]
~A3 B3|gfe fdB|AFA B2c|dfe dcB|\
~A3 ~B3|efe efg|faf gfe|1 dfe dcB:|2 dfe dBA|]
fAA eAA|def gfe|fAA eAA|dfe dBA|\
fAA eAA|def gfe|faf gfe|dfe dBA:|
```

=

.WAV Files

Offset	Size	Description	Value
0	4	Chunk ID	RIFF
4	4	Chunk data size	8
8	4	RIFF type	WAVE

Offset	Size	Description	Value
12	4	Chunk ID	“fmt”
16	4	Chunk Data Size	16 + *
20	2	Compression code	Int
22	2	Number of channels	Int
24	4	Sample rate	Hex
28	2	Block align	Hex
32	2	Significant bits per sample	Int
34	2	Extra format bytes	Int

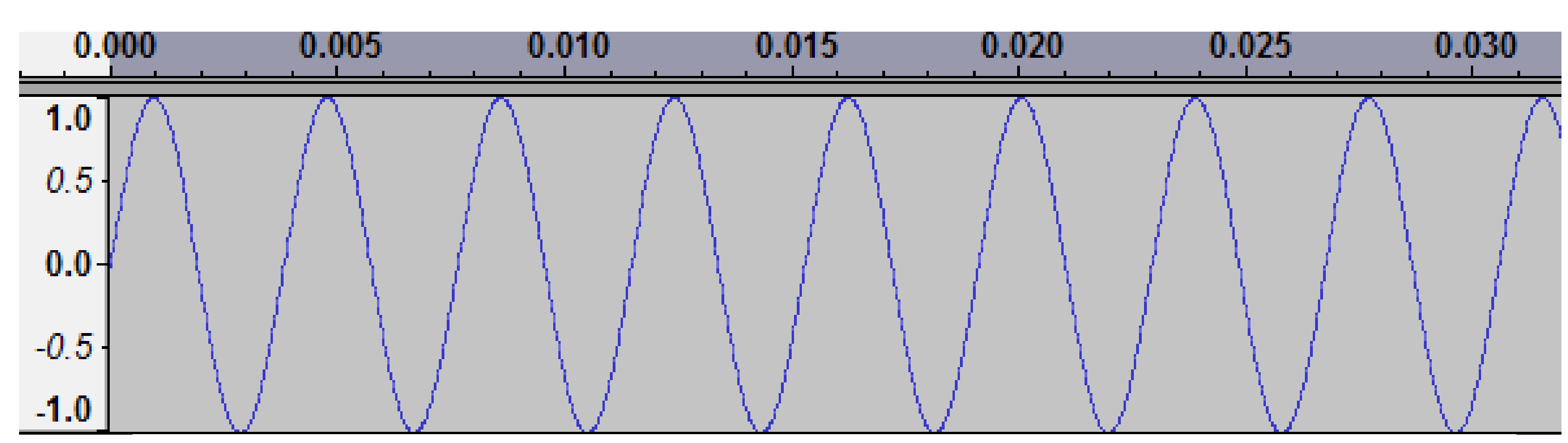
Offset	Length	Description	Value
36	4	Chunk ID	“data”
40	4	Chunk size	Depends on file
44	*	*	*

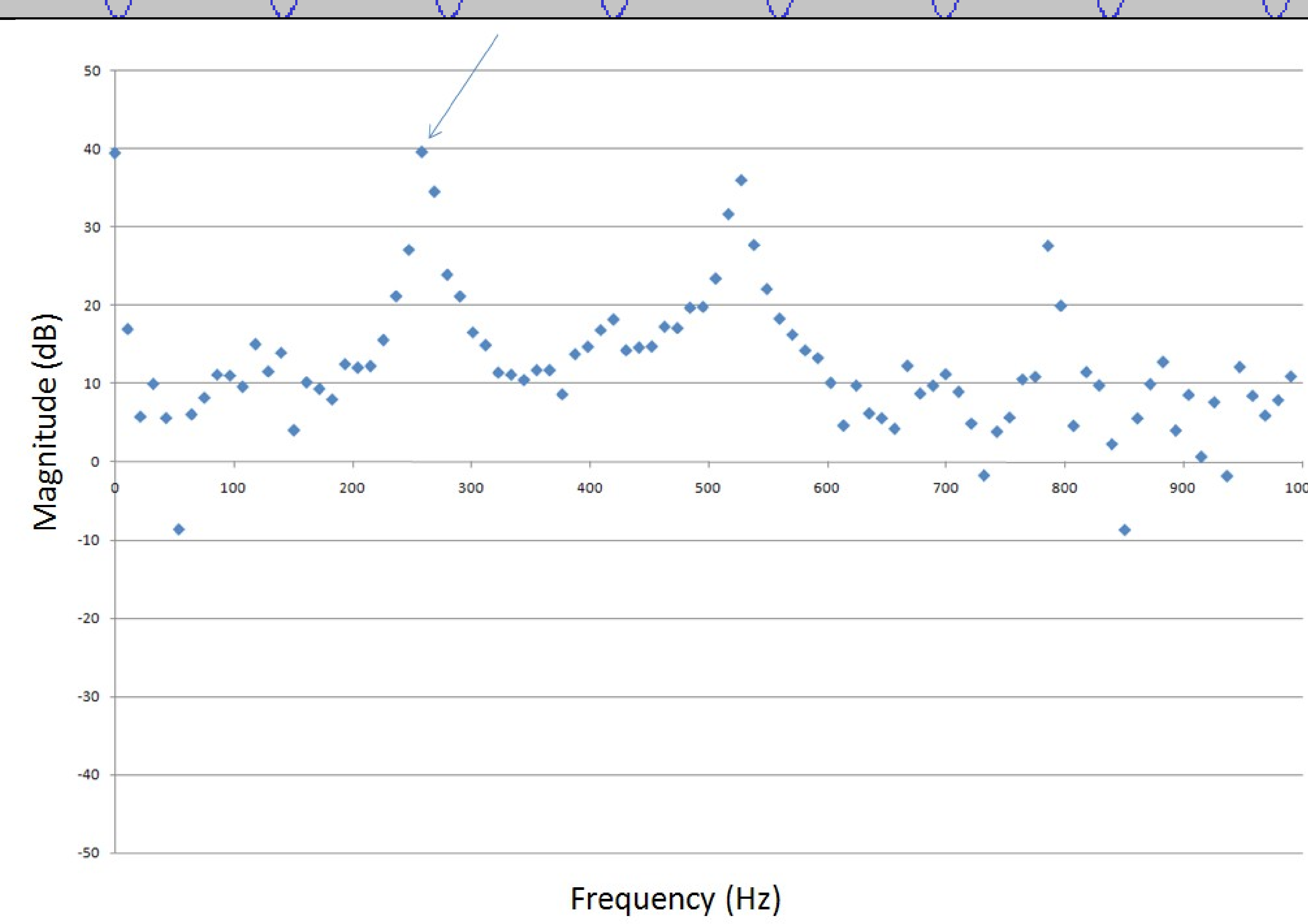
$$\hat{f}(\xi) = \int_{-\infty}^{\infty} f(x)\, e^{-2\pi i x \xi}\, dx,$$

$$\hat{f}(\xi) = \sum f(x)\cos(2\pi\xi x) + i\sum f(x)sin(2\pi\xi x)$$

To go from these sound waves into actual notes, a piece of rather complicated math is required - the Fourier transform. The Fourier transform, in sound, takes a function in the time domain, and changes it into a function in the frequency domain.

Middle C





Conclusion

This project successfully implemented the Fourier Transform to identify single notes. This procedure also works well for pure staccato music, in which notes do not overlap in time. For multiple notes, the Fourier Transform process will identify the various notes, but does not indicate which order the notes come in. To determine that, the .WAV file data must be used to find out when in time a new note begins. The Fourier Transform process would then be applied sequentially, each time using that portion of the .WAV file between where the current note starts and the next note starts. These start times are easy to determine for staccato music. But when notes overlap, it becomes more difficult. A possible procedure to do this was identified. A lot more could be done to extend this project. In the current version, it uses the Discrete Fourier Transform, which certainly gets the job done. However, it would be faster (and probably easier in the long run) to instead implement the Fast Fourier Transform, another algorithm that uses less samples and finds an answer more quickly. Another thing that needs to be implemented was the actual conversion to ABC format. The current program simply prints out the note it has decided on, and also three text files detailing the frequency, magnitude, and phase of each sample. Future implementations of this project could also figure out the relative lengths of each note, and use those to find the time signature and which length of note each is - quarter note, half note, sixteenth note, etc.