Luna Koizumi
Math 2270
April 25, 2012

.Wav Sound Compressions Using Maple

This project used Maple to compress .wav files of various instrument notes to see the effects compressions had on those notes.

What is compression? Compression is a method used to decrease data file sizes in order to save storage space or time it takes to save and open such data. Compression can be implemented on a variety of digital data including text files, images, and music. In text file compressions, the process can decrease its data size by up to 50% by inserting a character that identifies patterns. In images, there are two types of compressions. One is compression that leads to loss of data. Although an image size is decreased, it can also lead to the decrease in amount of data for the image, leading to decrease in quality. The other method of compression however, has the ability to decrease data size while allowing it to recover its original data even after compression has occurred.

There are at least four types of compression when it comes to music. Two that deal with sound volume and range, and the other two dealing with quality and size of music files. Dynamic range compression (DRC), also known as "compression" in the music industry is a method which reduces the volume of loud sounds and amplifies the volume of quiet sounds. Each sound has different levels of sound. For example, the beating of a drum most likely has a louder sound level than a person's whisper. In DRC, the engineer can change such audio file to make the drums quieter and the voice louder. Or, it can make the drums louder, also called "punchier" and the voice softer. In other words, it can change the volume of any sound in any combination. Some individuals claim that this process, although not apparent when listening to a compressed music file in the car or through an iPod, simply makes the music volume loud and does not preserve quality.

The other type of music compression is simply compressing the size of an audio file. This results in lossy or loseless compression (analogous to the losing and no loss of data in the image compressions). Examples of lossless compression are DTS-HD and Dolby True HD which retains all of its original information.

Although these four types of compression are categorized differently, they come hand in hand. In other words, DRC can also lead to smaller data sizes, and lossy compression.

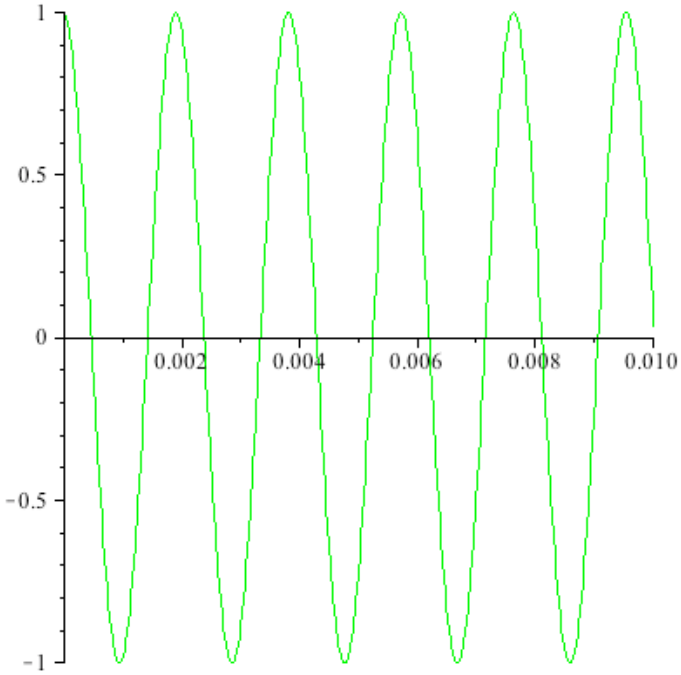
In this project, we focused on compressing sounds files. In order to do this, we decreased the dynamic range of the audio signals in the different sound files by eliminating the sounds the were present in the files that were very minor and outliers which when deleted, would not effect the quality of the sound to a listener's ear, greatly.

The Project:
I used a code that was provided to me through Dr. Gustafson's website to compress .wav files in Maple.

Luna Koizumi
Math 2270
April 25, 2012
First, I compressed a puretone:

```
> with(DiscreteTransforms): with(AudioTools): with(LinearAlgebra):
> compress := proc (x, z) if |(Re(z))| < x then return 0 else return Re(z) end if end proc:
> spectrum := proc (V, f) plot((Vector(Dimension(V), x→x)|map(x→Re(x), Vector(V, orientation=column))>,x=1..f)end proc;
#------------------------------------------------------
```

```
puretone := Vector(80000, x→ evalf(cos(2*261.626*2*Pi*x/44100)), datatype=float[8]):
Preview(Create(puretone), 0.. .01);
```



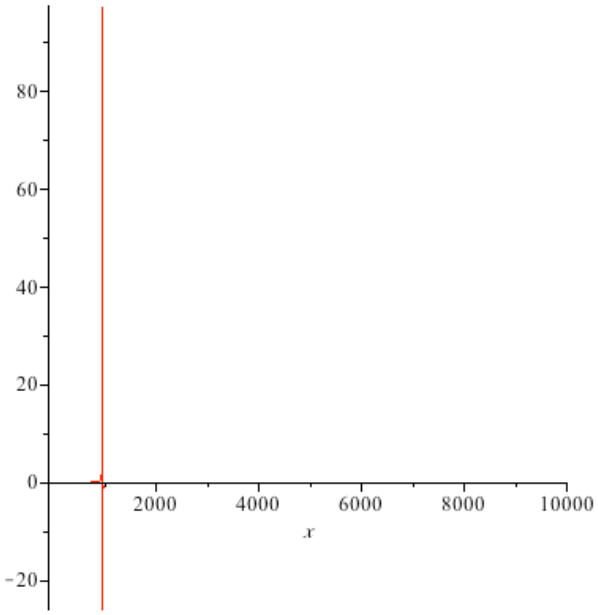
Graph 1

Here, cos(2*261.626*2*Pi*x/44100) describes the frequency of Graph 1 and through the graph, we can see that the amplitude is approximately 0.002 (=0.0034-0.0014). Here, the pure tone was created using a cosine equation.

The Fourier theorem states that we can describe any periodic waveform by the sum of a series of sine waves with frequencies in a harmonic series. Below, we change the pure tone data as a function of time as function of frequency.

```
Transformedpuretone:=FourierTransform(puretone):
Spectrum(transformedpuretone, 10000);
```

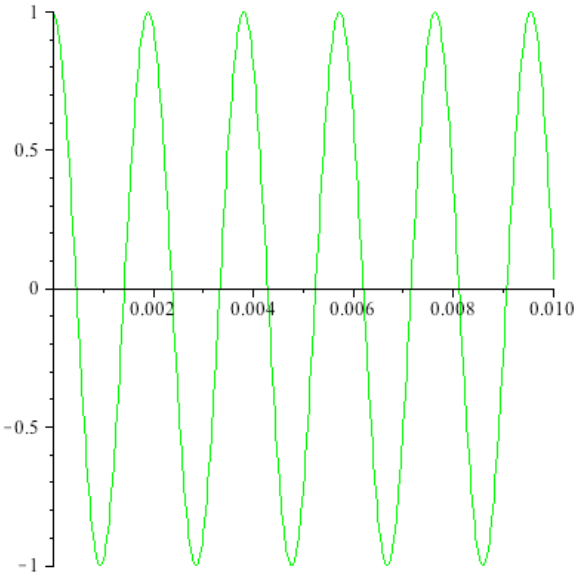
Luna Koizumi
Math 2270
April 25, 2012



Graph 2

Graph 2 shows the energy of pure tone. According to Graph 2, all the energy is at around 1000Hz (where the red spike is), which makes sense because in pure tone, only one tone sticks.
Likewise, we can verify this pure tone data by importing an audio file of a pure tone than an equation. In other words, we illustrate the same pure tone using an actual audio file of pure tone.

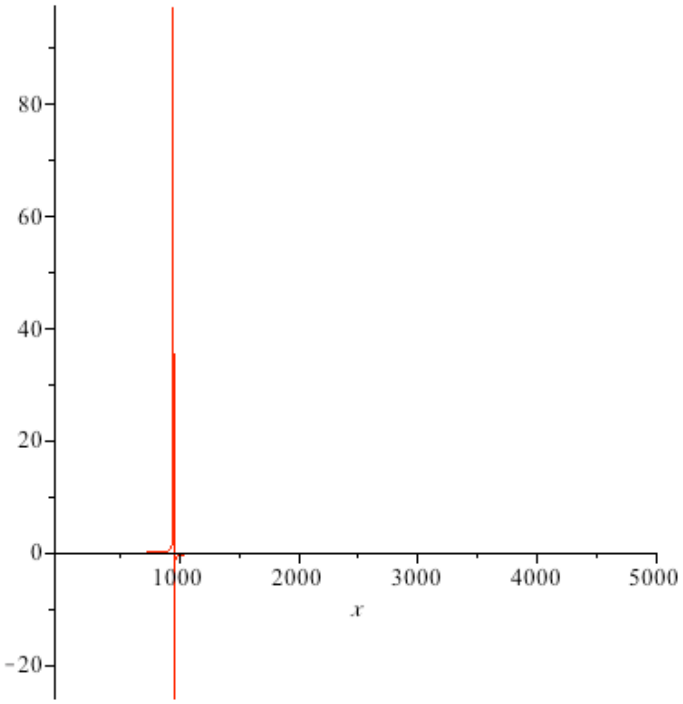
```
Audio:=Read("/u/class/k/c-kilr/puretonedata.wav")
Preview(audio, 0...01);
```



Graph 3

Luna Koizumi
Math 2270
April 25, 2012
Here, we observe that Graph 1 and Graph 3 is identical as it should be since we are working on the same exact pure tone. Below, we take the audio and make a data set that is the transpose of the audio vector.

```
data:=Vector(audio)+ :
transformeddata:=FourierTransform(data):
spectrum(transformeddata, 5000);
```



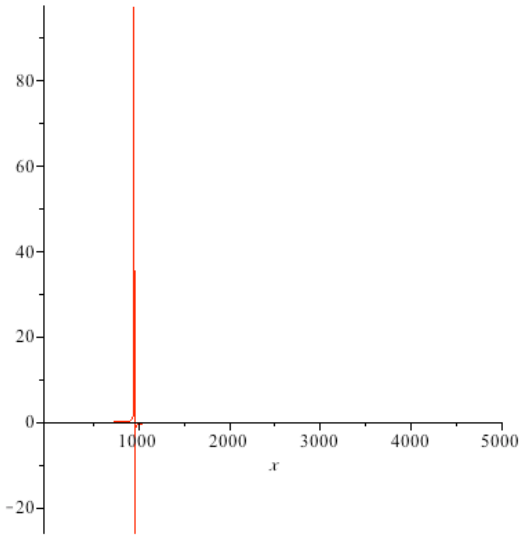
Graph 4

Graph 2 and Graph 4 is identical as it should be.

Now, we will compress the dataset and then see its energy frequencies after the compression.

```
compresseddata:=map(x→compress(.1,x), transformeddata):
spectrum(compresseddata,5000);
```

Luna Koizumi
Math 2270
April 25, 2012

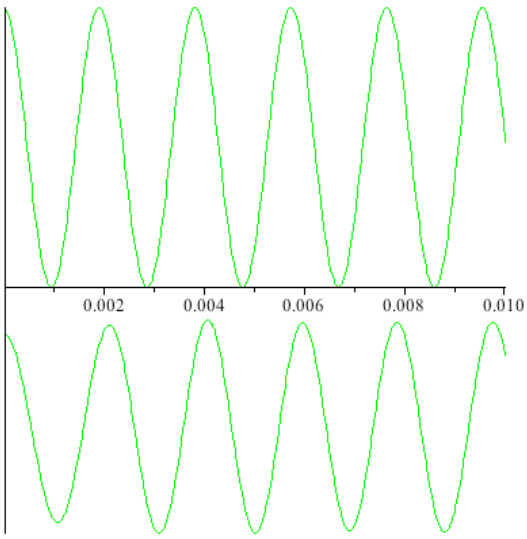


Graph 5

Graph 5 shows the energy spectrum after the compression has occurred on the pure tone audio file. Graph 4 and Graph 5 seems identical, at least to the human eye, as far as we know. This means that after the compression, nothing was lost and that the energy is still completely intact, at around 1000Hz as it was before the compression. This means that the compression, at least for the pure tone, did not lead to a loss in data.

Then, we inverse transform (Inverse Fourier Transform) the compressed data to get a sine wave. The Preview below allows us to see the sine wave from before the compression above the x-axis and the sine wave after the compression below the x-axis. That is why you see two plots on the same axis.

```
decodeddata:=map(x→Re(x), InverseFourierTransform(compresseddata)):
compressedaudio:= Create(decodeddata):
Preview(Create(<data|compressedaudio>),0.0..01);
```

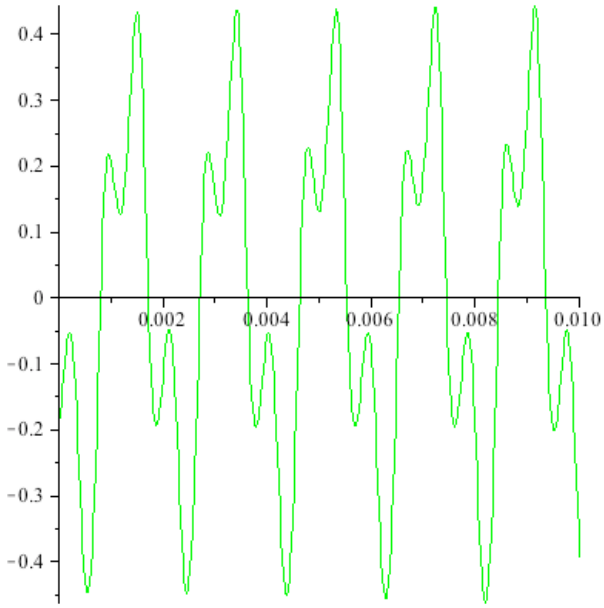


Graph 6

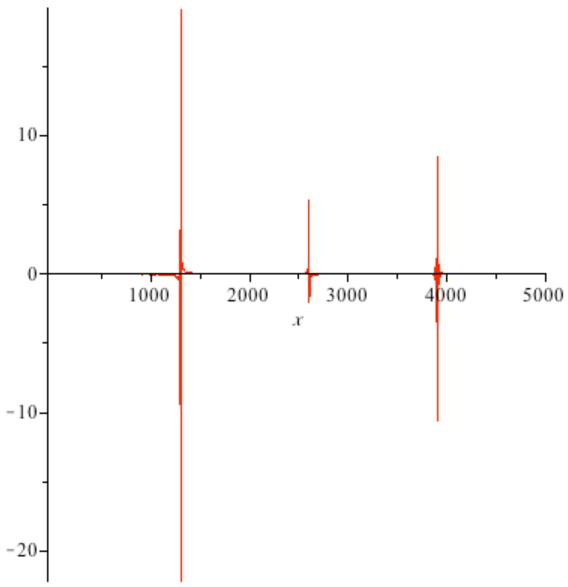
Luna Koizumi
Math 2270
April 25, 2012

As you can see in Graph 6, the sine wave seems almost identical. The only difference is that the amplitude seems a little smaller after the compression, but the period seems to remain the same. Again, we can come to a similar comparison as we did by comparing Graph 4 and 5, that it does not seem like the sound quality and data has remained intact.

Second, we will be compressing a note from a flute. We use the same Maple code as we did for the pure tone, except with a different input for the flute data. Before the compression, below was the sound wave and energy diagram:



Graph 7



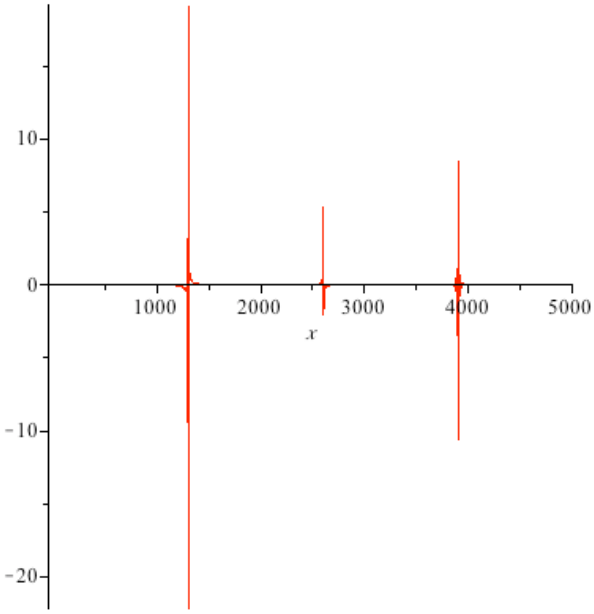
Graph 8

Here, the energy of the flute sound seems to be at three main points (corresponding to the spikes in Graph 8) at 1300Hz, 2600Hz, and 3900Hz. The flute has overtones unlike the pure tone which was just a steady sound with no overtones. This means that a lot of terms are needed to describe the sound graph. In the case of the flute audio, we need three terms of Fourier Series for the total energy.

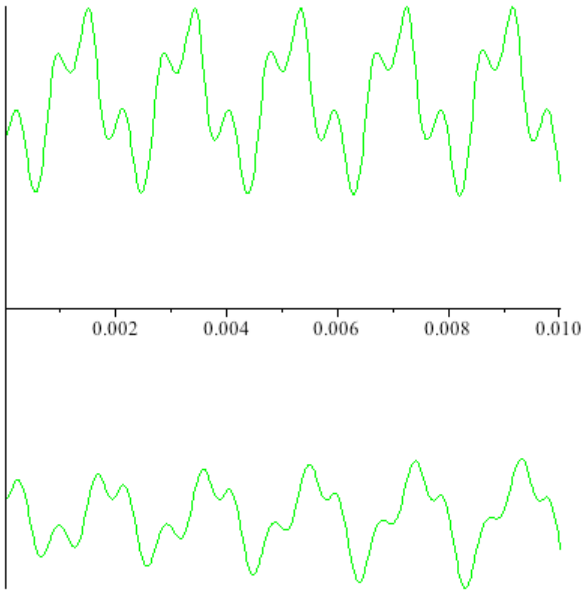
Now, take a look at Graph 9 and 10 below. Graph 9 is the energy graph after compression of the flute data, and Graph 10 is the comparison between the sound wave pre- and post-compression.

Graph 9 looks exactly the same as Graph 8, which implies that no data seems to be lost pre- and post- compression. When we look at Graph 9, we can see that the general shape of the sound waves are intact and the peaks seem to occur at similar times. However, one can argue that the sound ways do look different. Based on the patterns of the pre-compression sound wave, the second highest peak of each cycle in the pre-compression wave actually corresponds to the lowest peak of each cycle in the post-compression wave. However,

Luna Koizumi
Math 2270
April 25, 2012
because the main shape is still intact, we can make an educated guess that the sound will most likely sound the same to an individual's ear.

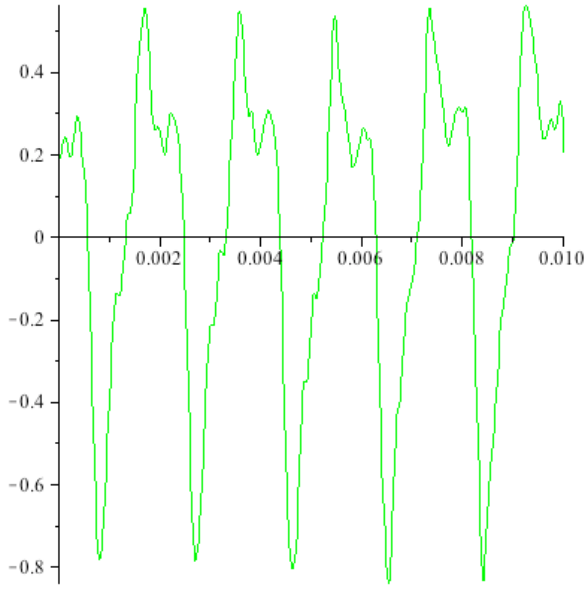


Graph 9

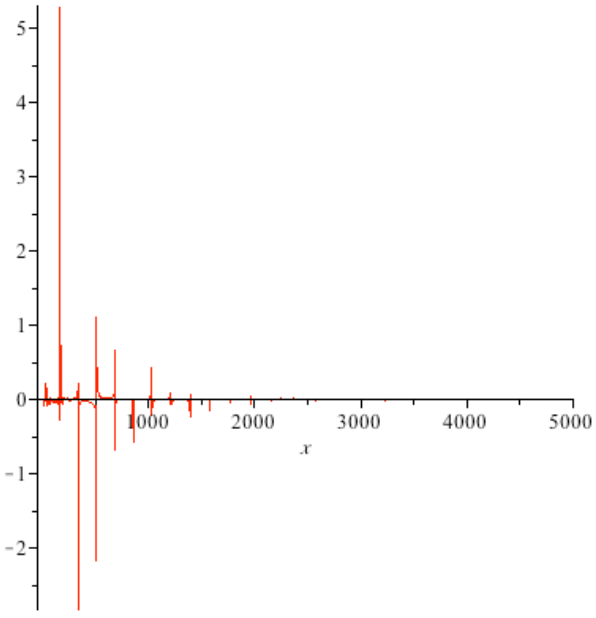


Graph 10

Next, we will analyze the piano! (This is where things get exciting!)



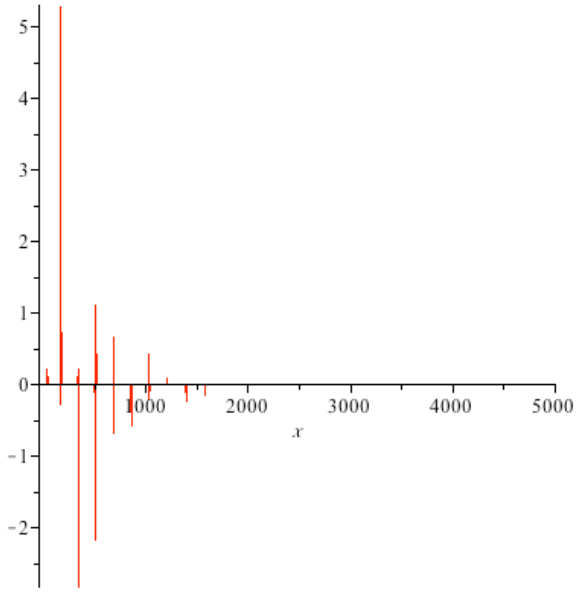
Graph 11



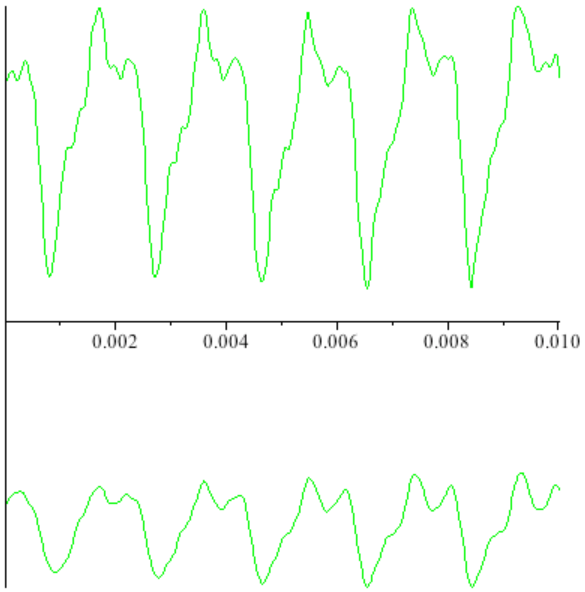
Graph 12

Here, we see the energies at various locations. There is a large energy frequency between 0 and 1000 Hz. After that, it becomes smaller and smaller and becomes "tally"-like from 1750Hz to approximately 3250 Hz, spreading out more and more at the energy level increases.

Luna Koizumi
Math 2270
April 25, 2012



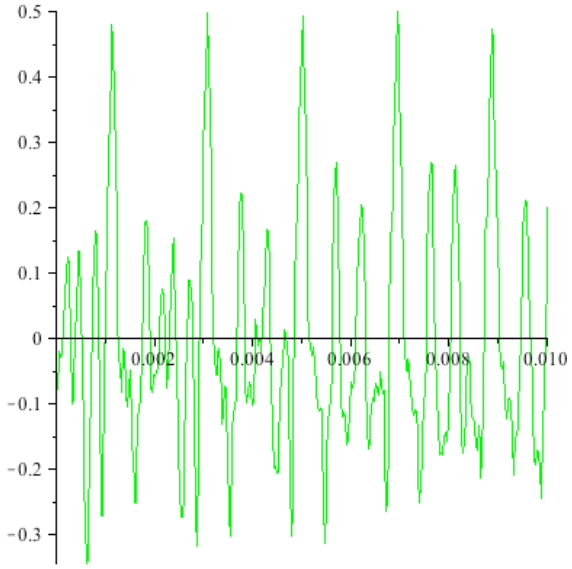
Graph 13



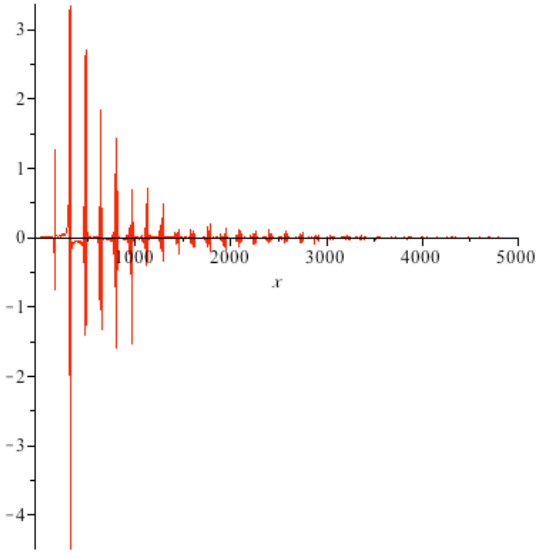
Graph 14

Graph 13 and 14 are the graphs after compression. From Graph 13, we see that the energy frequencies at higher energies have disappeared. In other words, we do not see any "tallies" past approximately 1570Hz. This tells us that the data for the energies above 1570 Hz has been eliminated from our audio data through compression. The before and after graph in Graph 14 is significantly more different than it has been for the other audio sounds. Here, the amplitudes are completely different and the "dips" and change in graph slopes are not as drastic. However, because the only data lost in the process was at higher Hzs, and is not a significant part of the audio, we hypothesize that the sound will sound similar to the ear after compression.

Next, we will analyze the trumpet's audio sound.



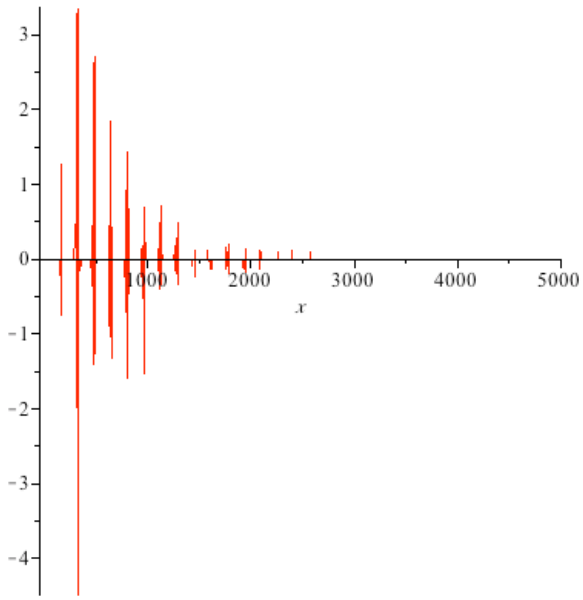
Graph 15



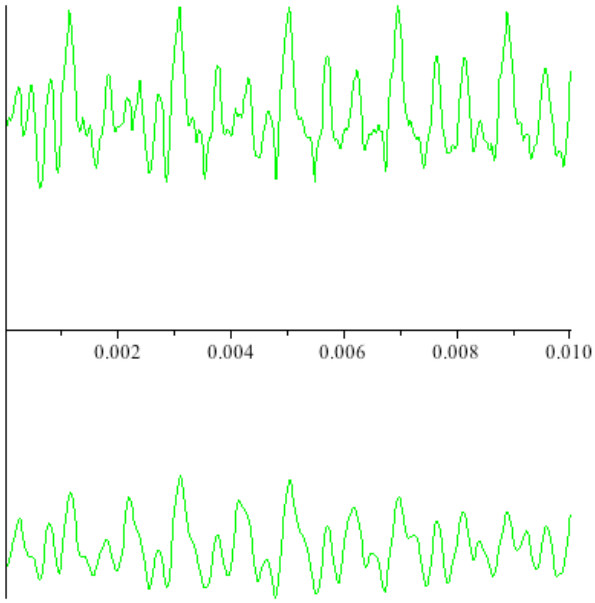
Graph 16

Luna Koizumi
Math 2270
April 25, 2012

The “before” graph (Graph 15 and 16) are quite distinct. The wave graph is very cluttered and seems to not have a perfect, repetitive shape from period to period. Graph 16 also covers ranges 0-5000Hz fully, which suggests a lot of overtone.



Graph 17



Graph 18

However, after compression, we do not see any energy after approximately 2600Hz in Graph 17. Like the piano model, higher energies were lost in the compression process. Graph 18 also has the most loss of original form that we have seen thus far. The “after” portion of Graph 18 does not have any sort of repetitive pattern, nor does it seem similar to the “before” portion of Graph 18.

It would be extremely interesting to re-convert the compressed audio data to an actual audio file and listen to it in comparison to the pre-compressed audio file. Unfortunately, for this project, I was not able to get the re-converting code to work in Maple. I could not get the .wav file to open up and play the re-written compressed audio to play. The code that I used was:

For the piano-

```
SoundFile := “/u/class/k/c-kilr/puretonedata.wav”;
Write(SoundFile, compressedaudio3);
process[launch](cat("sndrec32.exe ", SoundFile))
end proc;

PlayWave(compressedaudio3)
```

Luna Koizumi
Math 2270
April 25, 2012
If there was more time, it would have been interesting to see how much compression was too much. In this project, I used 10% compression, which seemed to work fairly well in maintaining its original audio sound. However, what if the compression was 20%? 50%? Better yet, how would those even sound like?