Introduction

The goal of my final project was to further explore the applications of the Fourier Transform beyond the scope of what we covered in class. Particularly, I was keenly interested in performing audio analysis of different instruments and/or vocals, visualizing their wave plots and spectrograms, and audiofying astronomical sources such as gravitational waves. I applied different levels of Fourier filtering to my audio data by zeroing out "bins" with a very low coefficient with respect to the rest of the polynomial. I culminated this project by writing a function to classify audio pitches to one of three notes: A4, F3, and F4; within this subpart of the project the main claim-to-fame of the Fourier transform became apparent.

I'll preface the rest of this report by mentioning that from my experience through this project, I've learned that the Fourier Transform isn't necessarily the best tool for audio analysis, depending on what you're doing and what type of filtering you're employing. Throughout this project I filtered using the zeroing bins strategy, and this doesn't really work when trying to clean noisy or static-y audio: this is why low pass, high pass, and bandpass filters exist. Noise not only adds a lot of extra data to the signal, it also amplifies the original signal present. In doing so, it cannot be completely removed by zeroing out bins in the frequency domain.

There are essentially three parts to this project:

1. Explore the Fourier Transform with instruments and audio samples, and plot spectrograms.
2. Audiofy gravitational waves.
3. Use the Fourier Transform to classify pitches to corresponding frequencies.

For each part, there is a Methodology section, followed by a Results section. At the end, there is an overall conclusion.

At the top of this notebook you'll several functions that are reused throughout to minimize cluttered code. Addtionally, here is a button to toggle on or off the code in the notebook. This was taken from a StackOverflow forum.

Out[122]: Click here to toggle on/off the raw code.

These are all of the packages used throughout the notebook.

This plotting function is used to make simple waveplots of audio so that the sound can be visualized on a time vs amplitude graph.

This spectrogram plotting function serves a simlar purpose as to the above function, but instead of plotting a simple time vs amplitude graph, it uses the librosa library to plot spectrograms.

This function was used to plot filtered signals.

This function was used to audiofy data, and specifically tailored to audiofying the LIGO data from H1 at Hanford, WA and L1 at Livingston, LA.

This is the classifier. It can classify a given .wav file as one of three frequencies: A4, F3, or F4. It prints the classification of the .wav file, plots its spectrogram, and plots the spectrogram of the note it is classified as.

Methodology: Part 1

In this part, several .wav files pertaining to certain instruments were loaded in and analyzed. Fourier filtering was attempted to clean or alter these files in any way. Since these files were not noisy to begin with, Fourier filtering did very little, except to muffle the audio, effectively compressing the data and decreasing the fidelity.

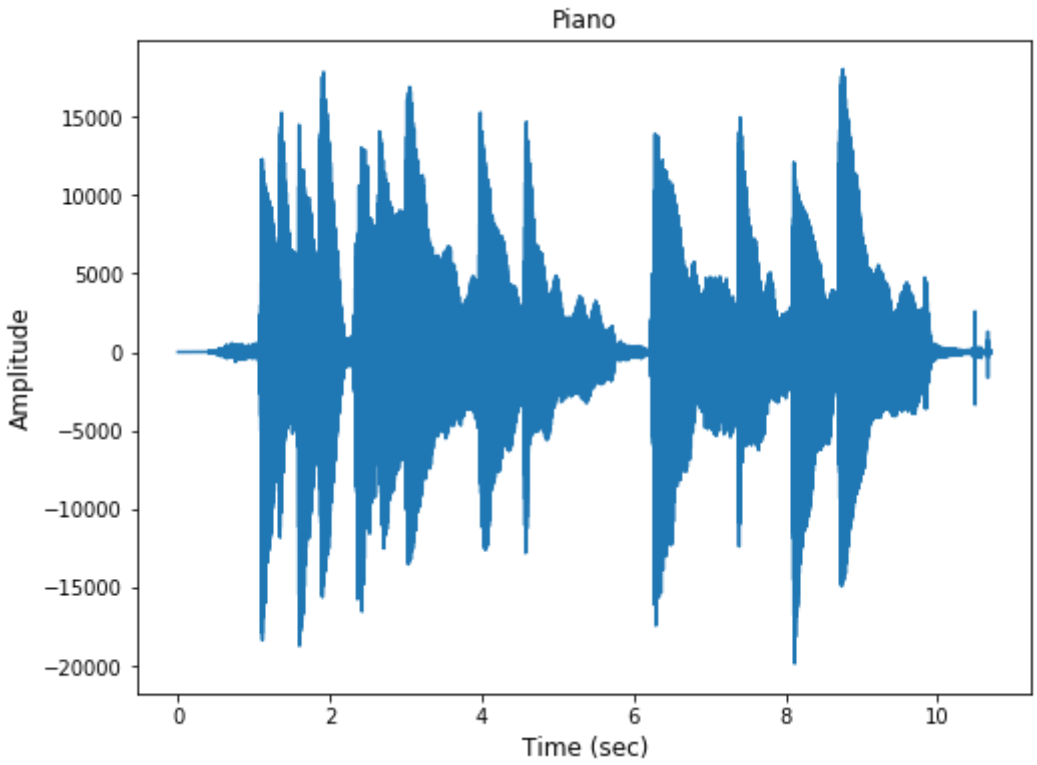
Pictured below is the first waveplot for a 10 second piano soundtrack. The audio can be listened to using the audio player. Immediately following is the piano sound converted to the frequency domain via a FFT. Due to the Fourier Transform being over such a long period of time, there are a lot peaks, making filtering by zeroing out bins very difficult. This is not something that was realized in the methodology until the very end, otherwise shorter audio samples would have been used to demonstrate the effect of filtering more profoundly.

Nonetheless, the filtered waveplot is pictured below as well. As is apparent, there is very little change. This is for two reasons:

1. Only very low coefficients in the frequency domain were zeroed out
2. The audio is already quite clean, so there is very little noise or imperfection to actually filter. Doing so is like grasping at air.

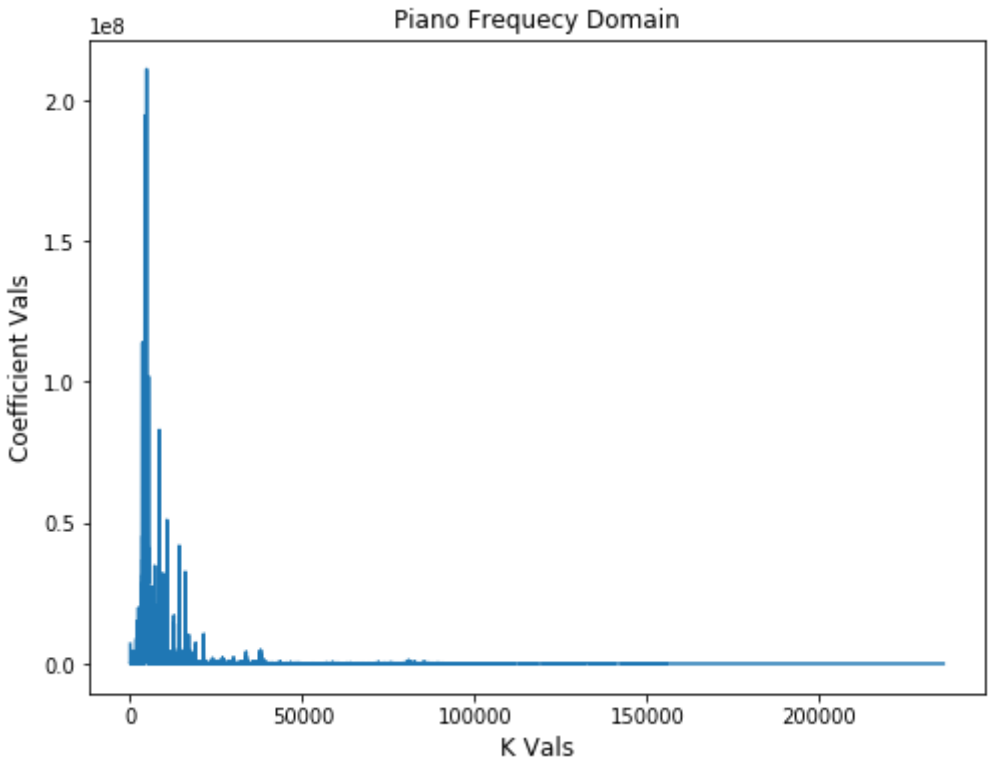
Following this, the rest of the waveplots and respective filtering are illustrated: Violin, Guitar, Vocal, Synth.

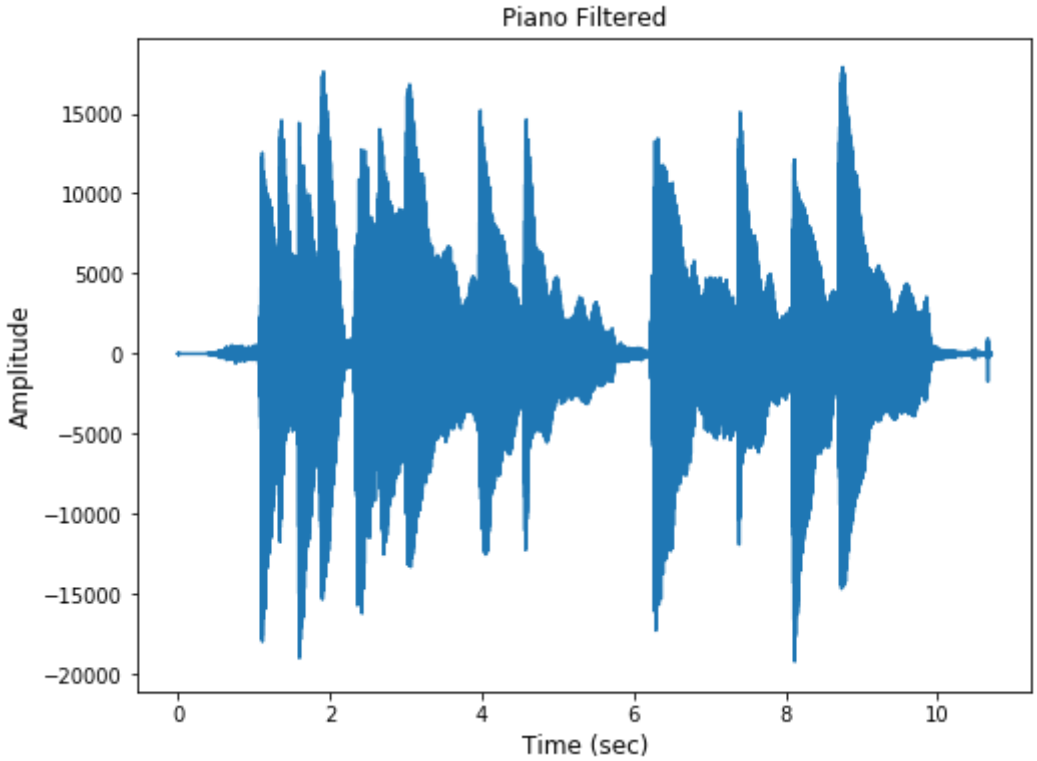

0:00 / 0:10


Piano
Amplitude
15000
10000
5000
0
−5000
−10000
−15000
−20000
0
2
4
6
8
10
Time (sec)

Out[130]:

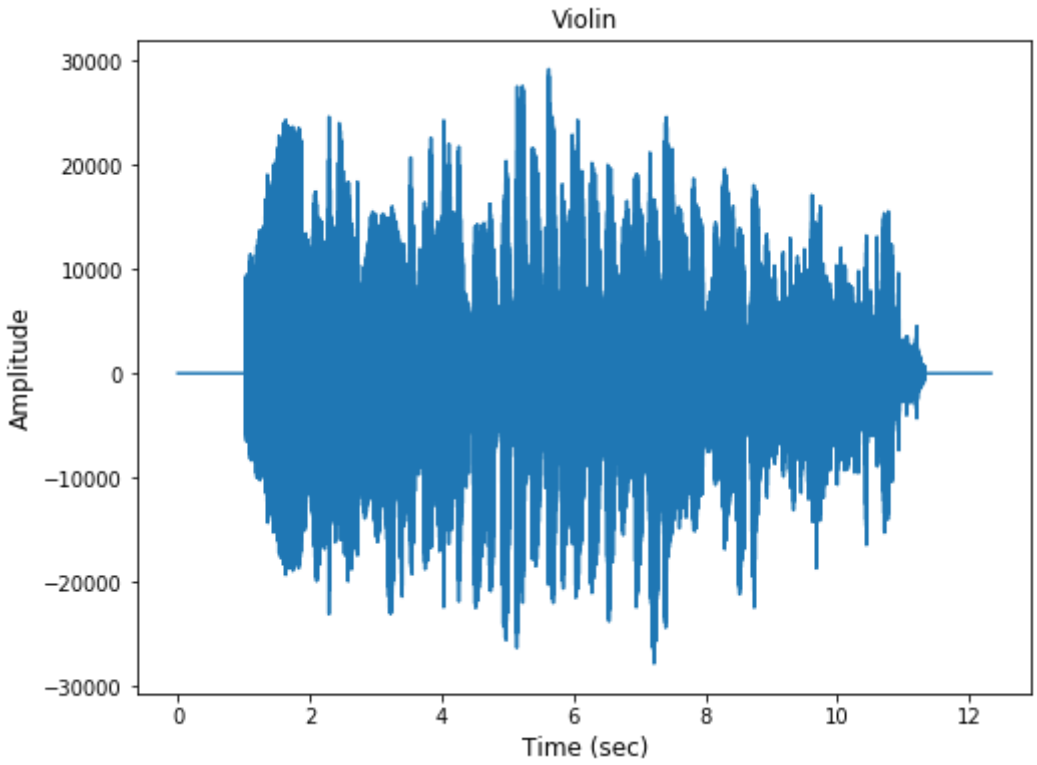

0:00 / 0:10


Piano Frequecy Domain
1e8
2.0
1.5
1.0
0.5
0.0
Coefficient Vals
0
50000
100000
150000
200000
K Vals


Piano Filtered
15000
10000
5000
0
−5000
−10000
−15000
−20000
Amplitude
0
2
4
6
8
10
Time (sec)

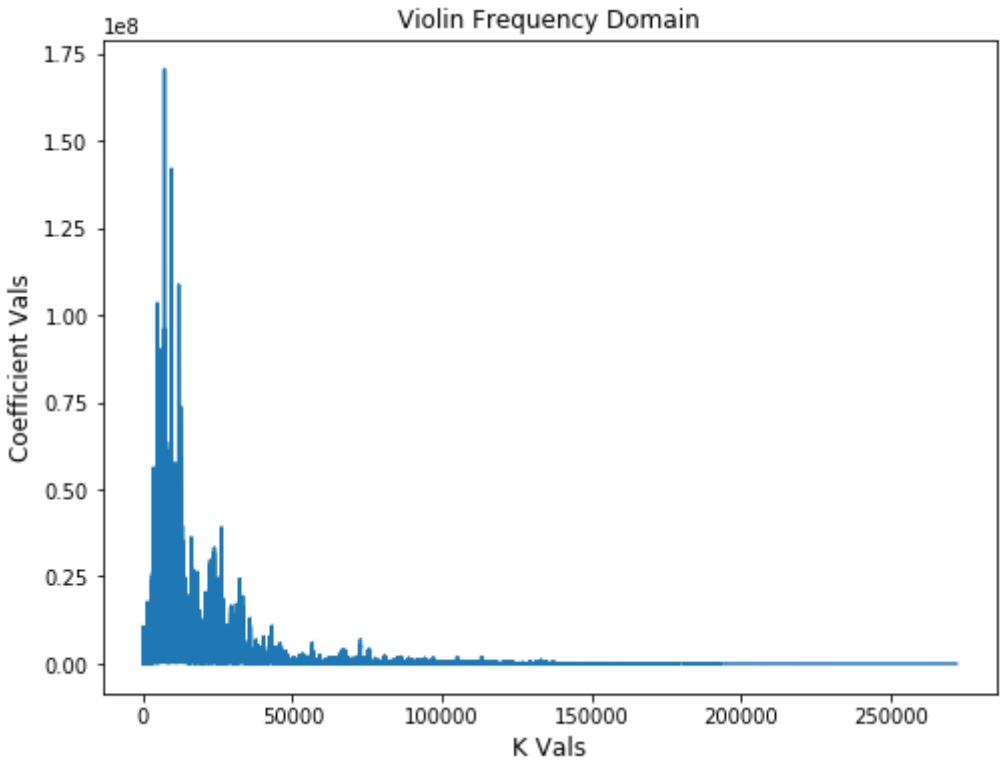
Out[131]:

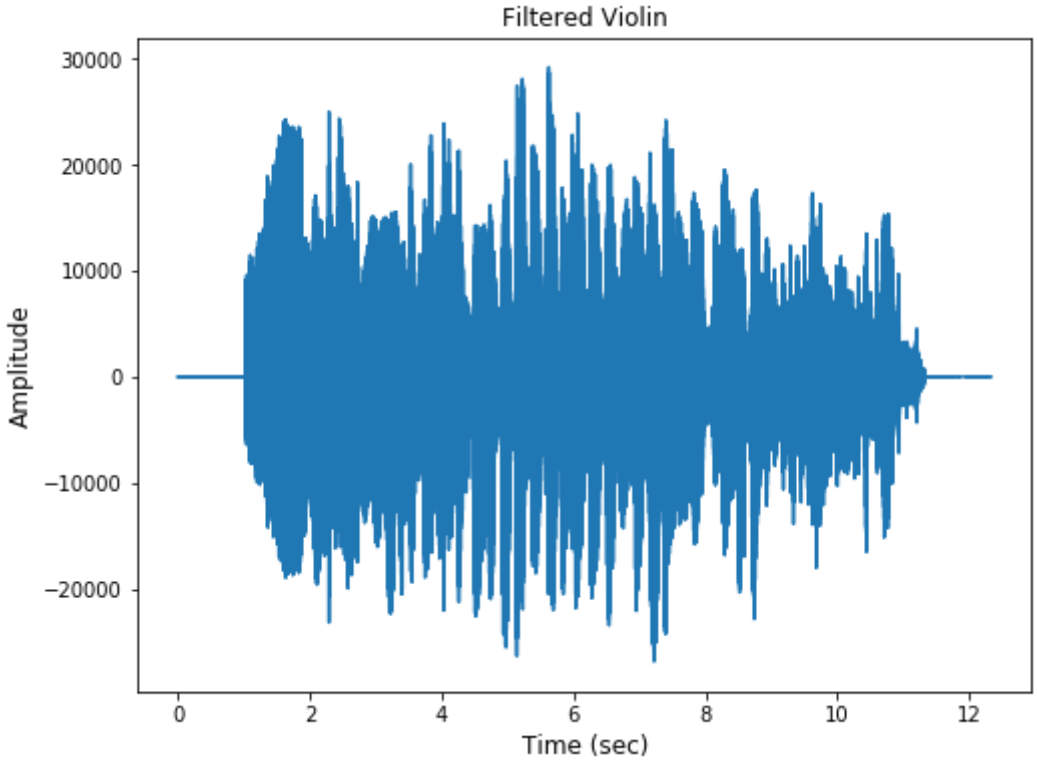

0:00 / 0:12


Violin
Amplitude
Time (sec)
30000
20000
10000
0
−10000
−20000
−30000
0
2
4
6
8
10
12

Out[132]:

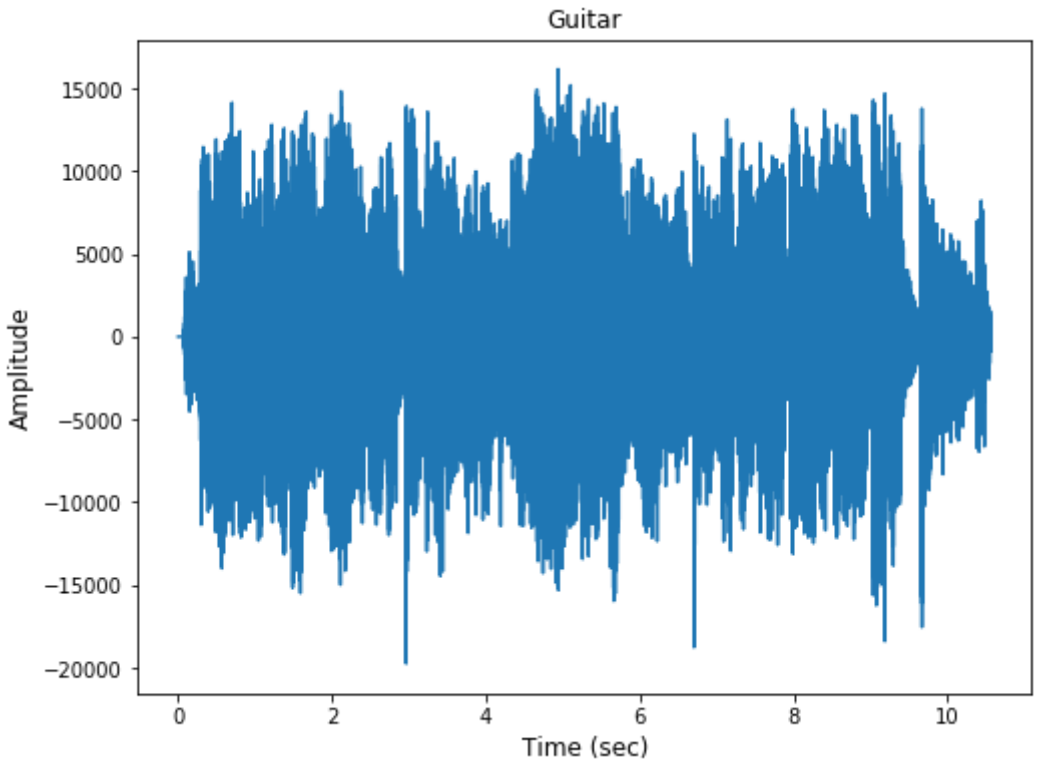

0:00 / 0:12


Violin Frequency Domain
1e8
1.75
1.50
1.25
1.00
0.75
0.50
0.25
0.00
Coefficient Vals
0
50000
100000
150000
200000
250000
K Vals


Filtered Violin
30000
20000
10000
0
−10000
−20000
Amplitude
0
2
4
6
8
10
12
Time (sec)

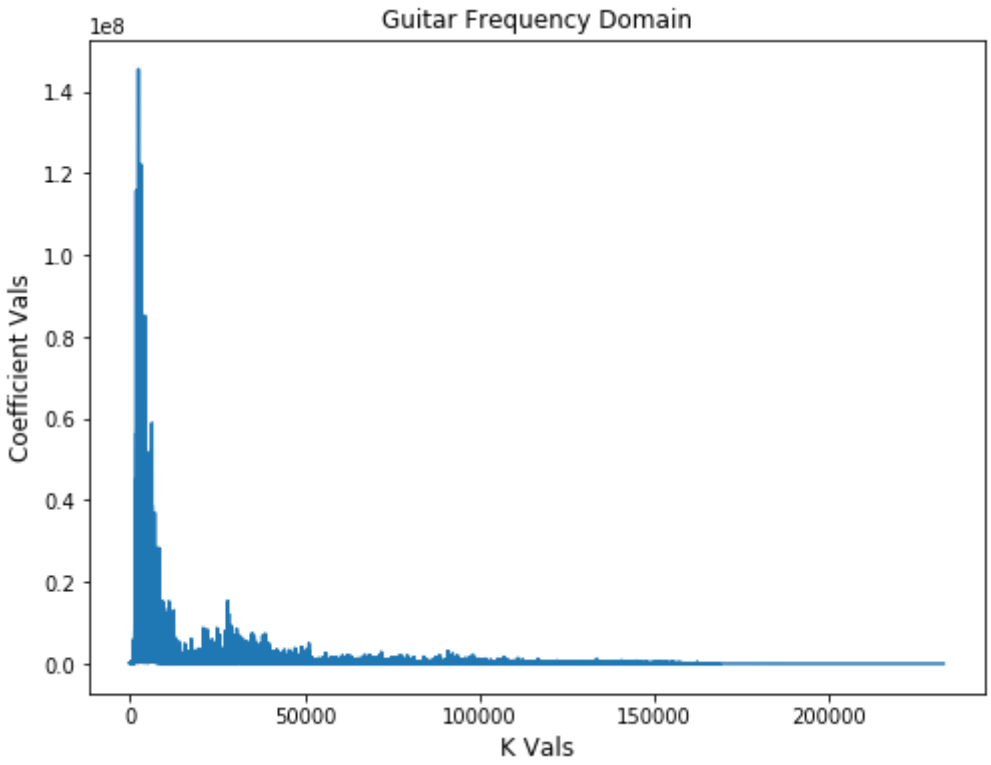
Out[133]:

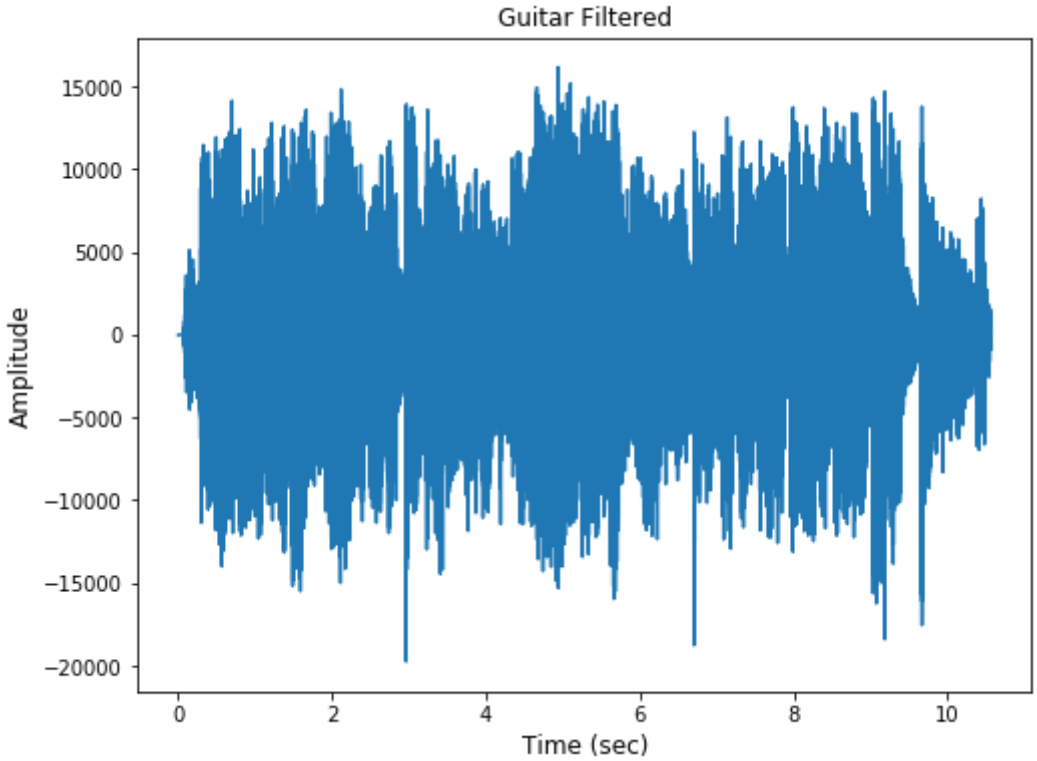

0:00 / 0:10


Guitar
Amplitude
Time (sec)
15000
10000
5000
0
−5000
−10000
−15000
−20000
0
2
4
6
8
10

Out[134]:

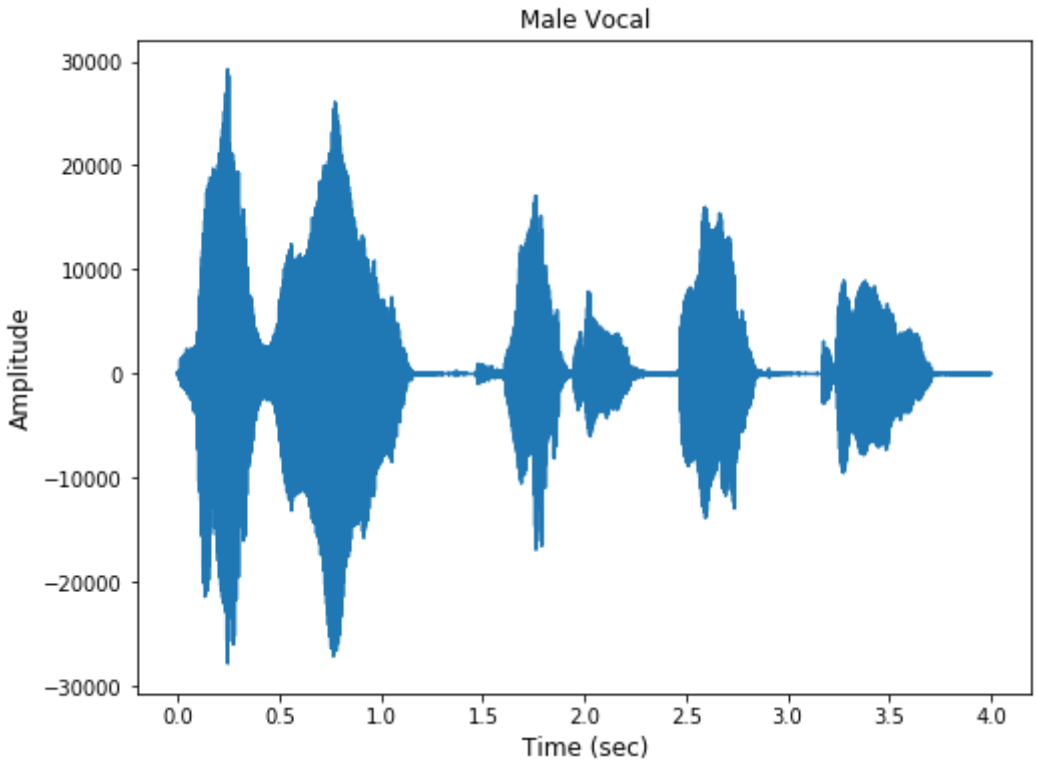

0:00 / 0:10


Guitar Frequency Domain
1e8
1.4
1.2
1.0
0.8
0.6
0.4
0.2
0.0
Coefficient Vals
0
50000
100000
150000
200000
K Vals


Guitar Filtered
15000
10000
5000
0
−5000
−10000
−15000
−20000
Amplitude
0
2
4
6
8
10
Time (sec)

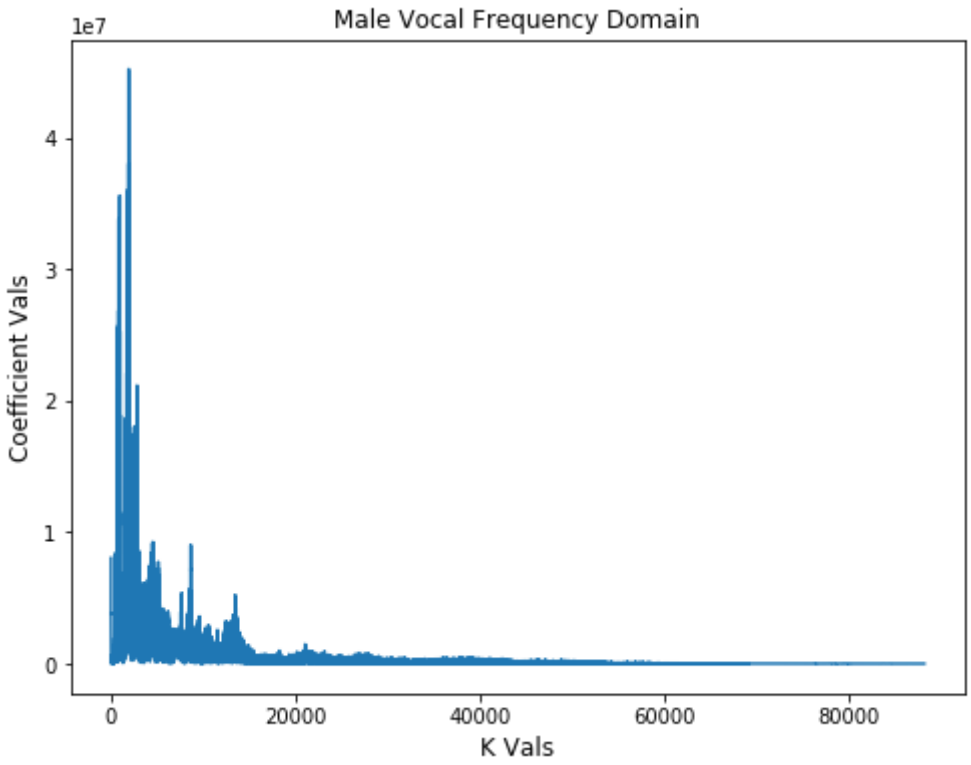
Out[135]:

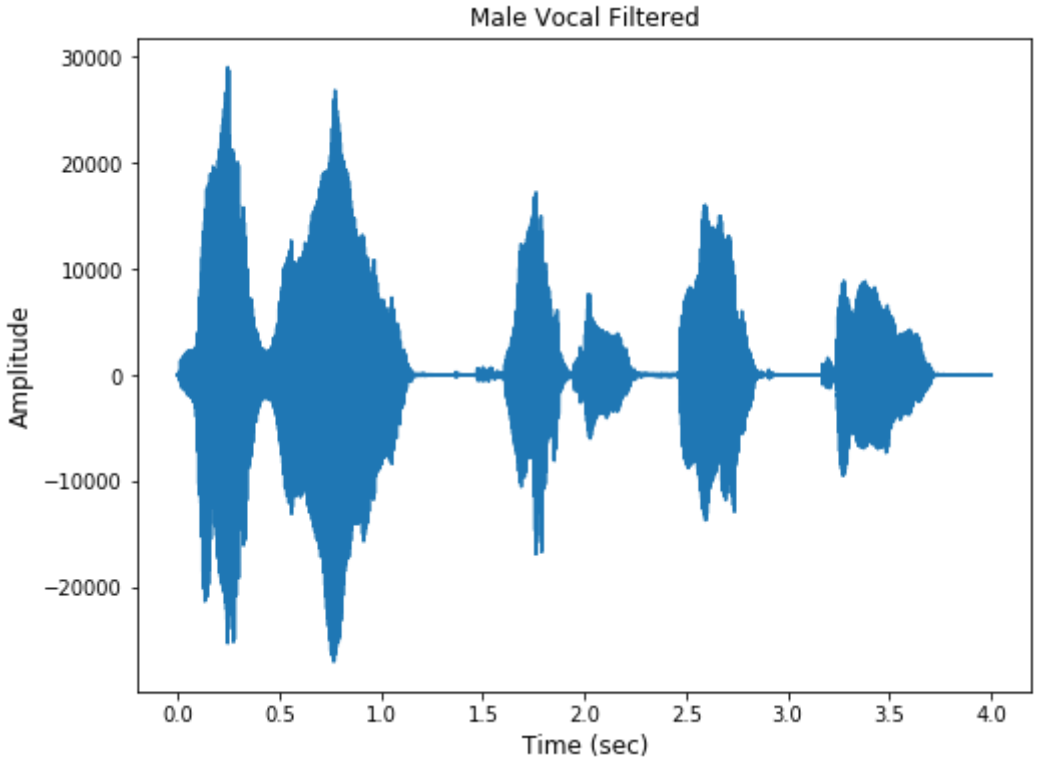

0:00 / 0:04


Male Vocal
Amplitude
Time (sec)
30000
20000
10000
0
−10000
−20000
−30000
0.0
0.5
1.0
1.5
2.0
2.5
3.0
3.5
4.0

Out[136]:

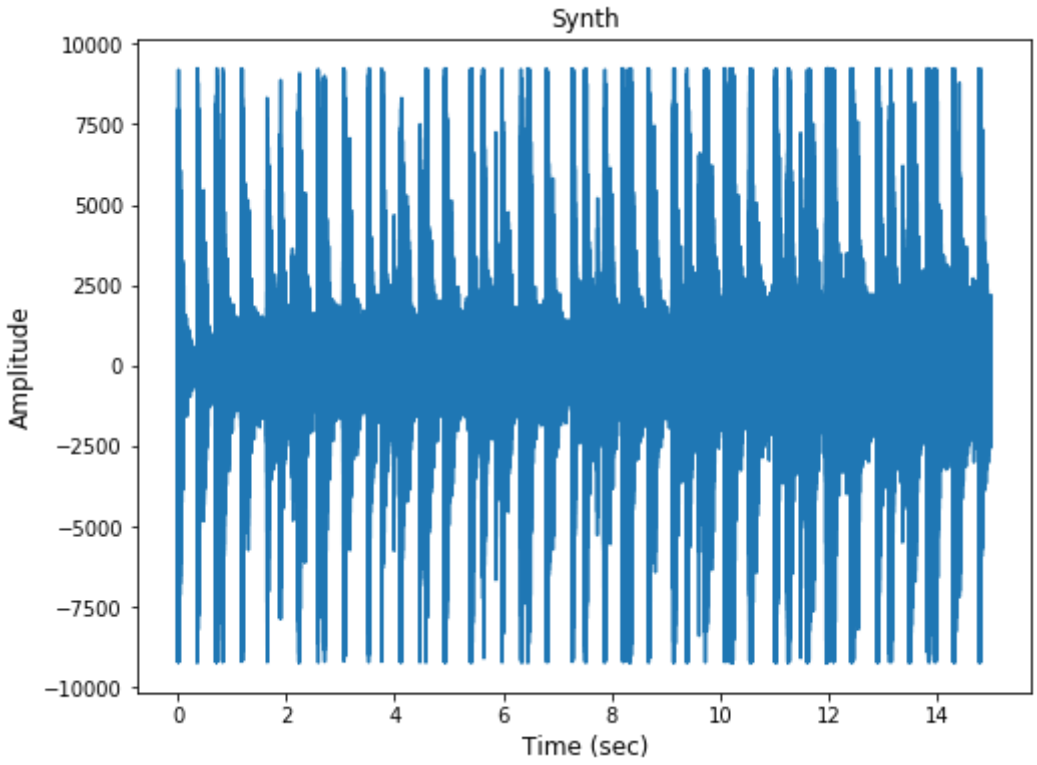

0:00 / 0:04


Male Vocal Frequency Domain
1e7
Coefficient Vals
K Vals
0
1
2
3
4
0
20000
40000
60000
80000


Male Vocal Filtered
Amplitude
Time (sec)
30000
20000
10000
0
−10000
−20000
0.0
0.5
1.0
1.5
2.0
2.5
3.0
3.5
4.0

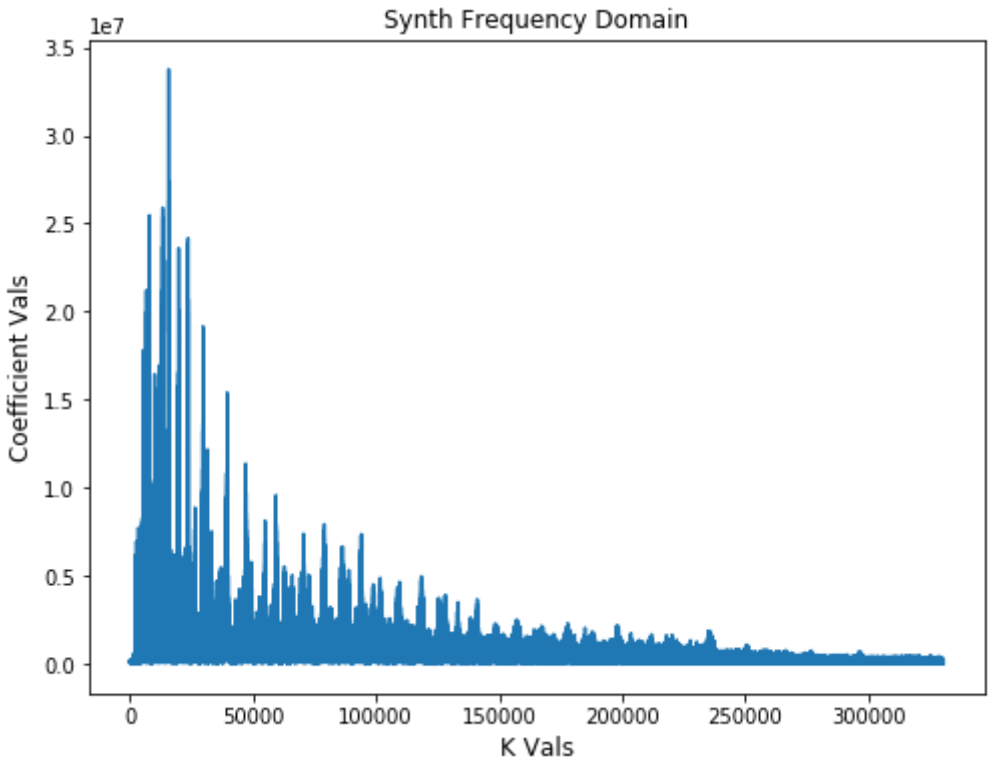
```
/usr/local/lib/python3.6/site-packages/scipy/io/wavfile.py:273: WavFile
Warning: Chunk (non-data) not understood, skipping it.
  WavFileWarning)
```

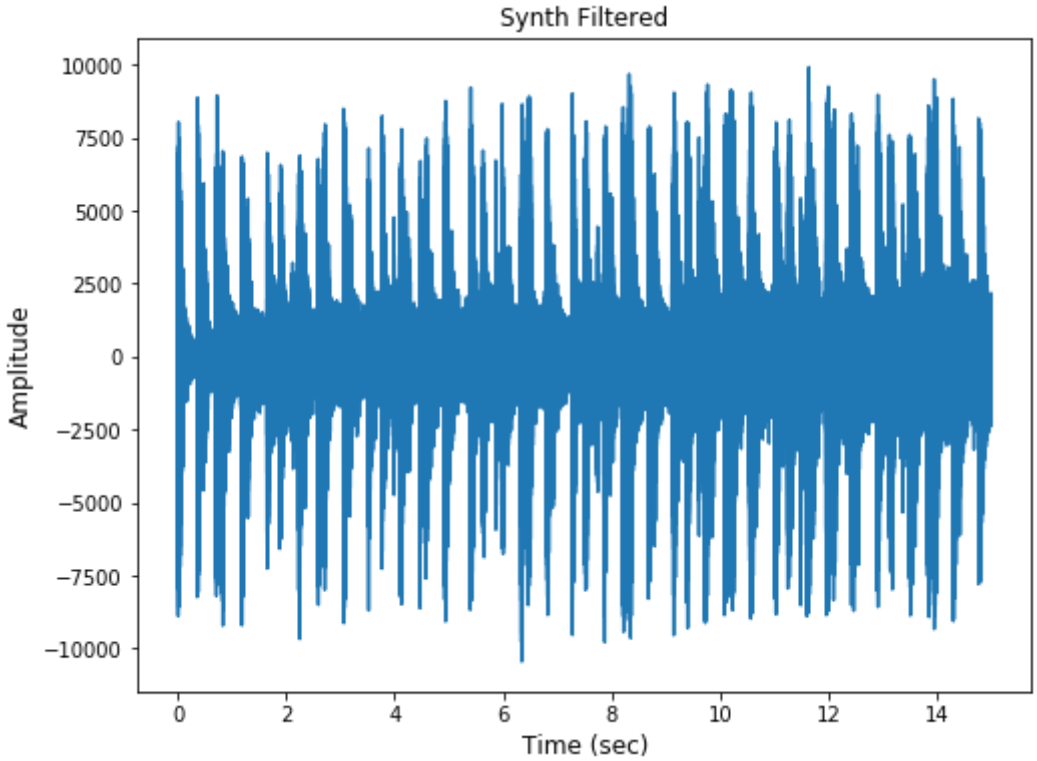
Out[137]:



Out[138]:

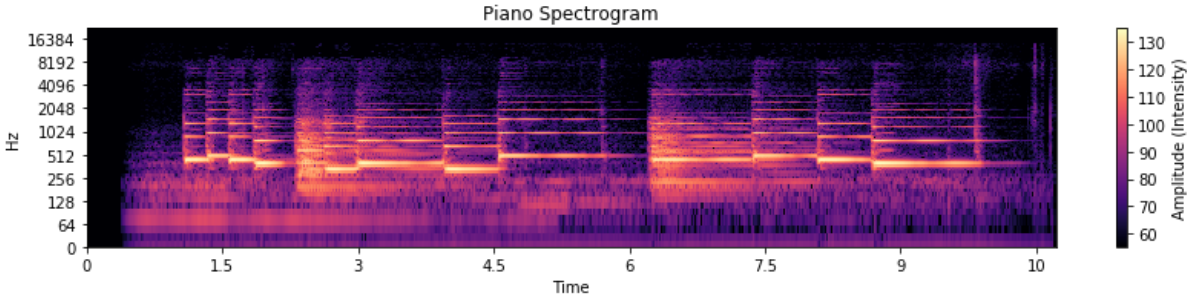

0:00 / 0:15

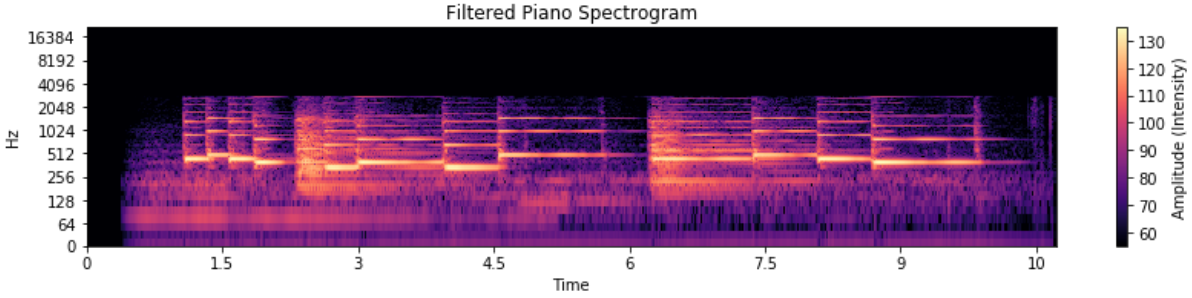

Synth Frequency Domain
1e7
3.5
3.0
2.5
2.0
1.5
1.0
0.5
0.0
Coefficient Vals
0
50000
100000
150000
200000
250000
300000
K Vals

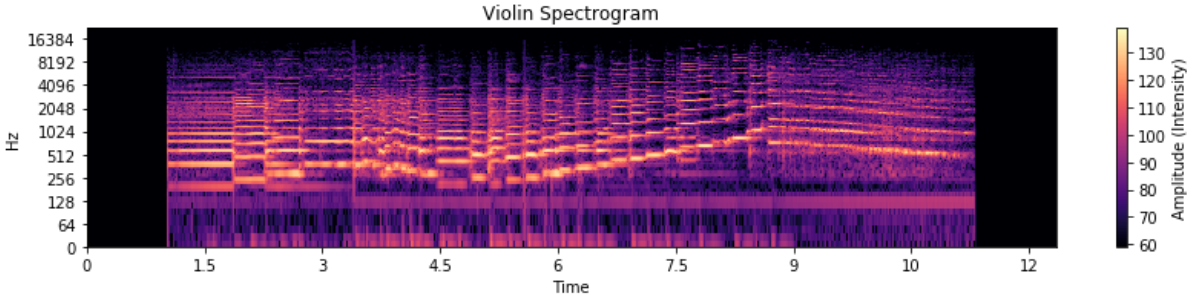

Synth Filtered
10000
7500
5000
2500
0
−2500
−5000
−7500
−10000
Amplitude
0
2
4
6
8
10
12
14
Time (sec)

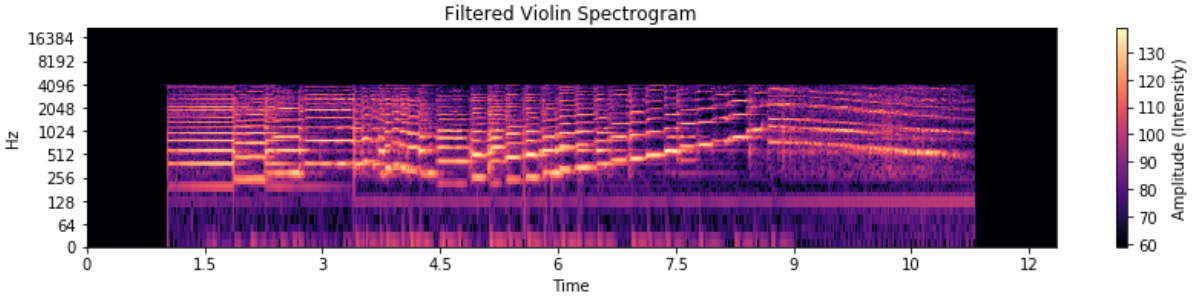
As is clear, different instruments and sounds have different waveplots. The way in which the instrument is played or the sound is delivered has an inherent effect on these waveplots as well. Compared to the piano and vocals, the other sounds were more closely packed together. Of particular note is the synth, however. Being a digital sound, it is a very controlled, symmetric waveplot.

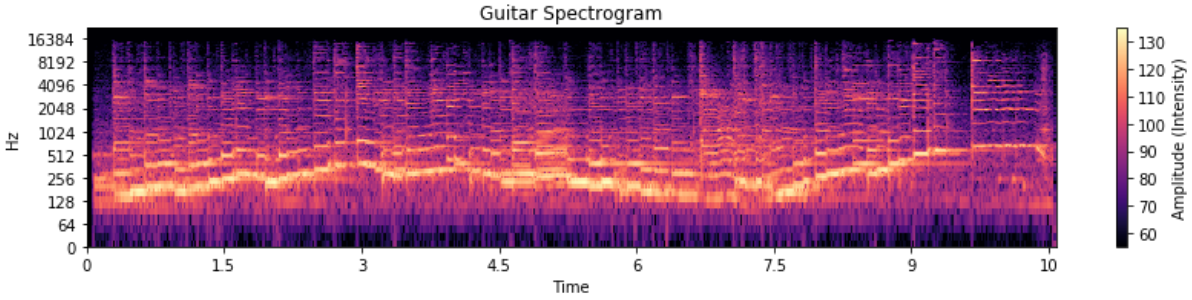
However, these waveplots give very little information as to what the actual instrument sounds like. To visualize sound -- so to speak -- pictured below are the spectrograms of the various instruments and sounds, as well as their filtered counterparts.

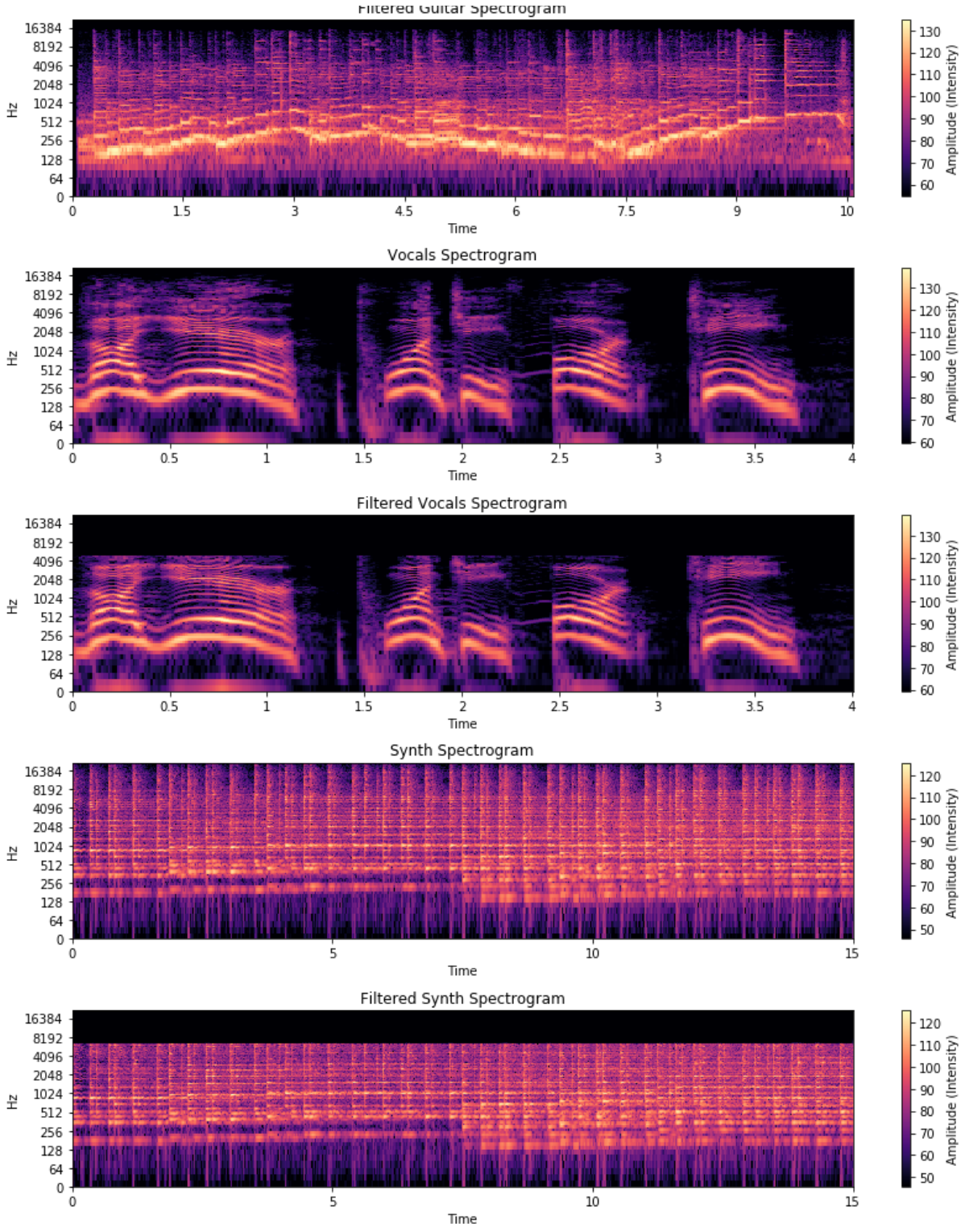

Piano Spectrogram
Hz
16384
8192
4096
2048
1024
512
256
128
64
0
0
1.5
3
4.5
6
7.5
9
10
Time
Amplitude (Intensity)
130
120
110
100
90
80
70
60


Filtered Piano Spectrogram
Hz
16384
8192
4096
2048
1024
512
256
128
64
0
0
1.5
3
4.5
6
7.5
9
10
Time
Amplitude (Intensity)
130
120
110
100
90
80
70
60


Violin Spectrogram
Hz
16384
8192
4096
2048
1024
512
256
128
64
0
0
1.5
3
4.5
6
7.5
9
10
12
Time
Amplitude (Intensity)
130
120
110
100
90
80
70
60


Filtered Violin Spectrogram
Hz
16384
8192
4096
2048
1024
512
256
128
64
0
0
1.5
3
4.5
6
7.5
9
10
12
Time
Amplitude (Intensity)
130
120
110
100
90
80
70
60


Guitar Spectrogram
Hz
16384
8192
4096
2048
1024
512
256
128
64
0
0
1.5
3
4.5
6
7.5
9
10
Time
Amplitude (Intensity)
130
120
110
100
90
80
70
60


Filtered Guitar Spectrogram
Vocals Spectrogram
Filtered Vocals Spectrogram
Synth Spectrogram
Filtered Synth Spectrogram
Hz
Time
Amplitude (Intensity)

Results: Part 1

Although Fourier Filtering looked as if it was doing nothing, the spectrograms give a different picture. Filtering simply removed the less wanted frequencies from the audio signal. Most of the important frequencies for the majority of the audio files are below 512 Hz, so chopping off the unimportant frequencies towards the top of the spectrogram doesn't necessarily ruin the important sounds, but rather it limits the "depth" of the sound, hence the muffled audio.

Additionally, waveplots didn't really provide much insight as to what the audio sounded like. With spectrograms, however, the dips and rises in frequency are clearly visible, as is the strength of a particular note. The piano spectrograms look as if someone is pressing keys at a set interval, causing sound to propagate. The guitar and violin spectrograms are much more unified, likely due to string instruments' sound oscillating and persisting longer (perceivably) in the air.

Now, the vocals are very interesting. Speech is caused by the lungs modulating air outwards, vibrating our voice box and causing changes in the air, resonating outwards. Humans aren't computers and can't necessarily control their voice perfectly, so it makes sense that vocals would appear rounded in the spectrogram, since we're constantly modulating the air to speak.

The synth is the exact opposite of the vocals: very straight and uniform, as expected from a digital sound. The shift in tone at the 7 second mark is clearly visible, however. The sound is constantly rising in this audio, except when it shifts lower for just a moment and then continues its upward trend.

Methodology: Part 2

The focus of this part was to take the gravitational waves data from LIGO and convert it to some sort of audio. This data is originally viewed as time vs strain. Now, the original data, plotted below, spans a period 0.2 ~ 0.21 seconds. This was elongated by creating a repeating signal, and done so because 0.2 seconds was not enough to properly listen to the audio. It was lengthed by 20 times to get approximatley 4 seconds of audio.

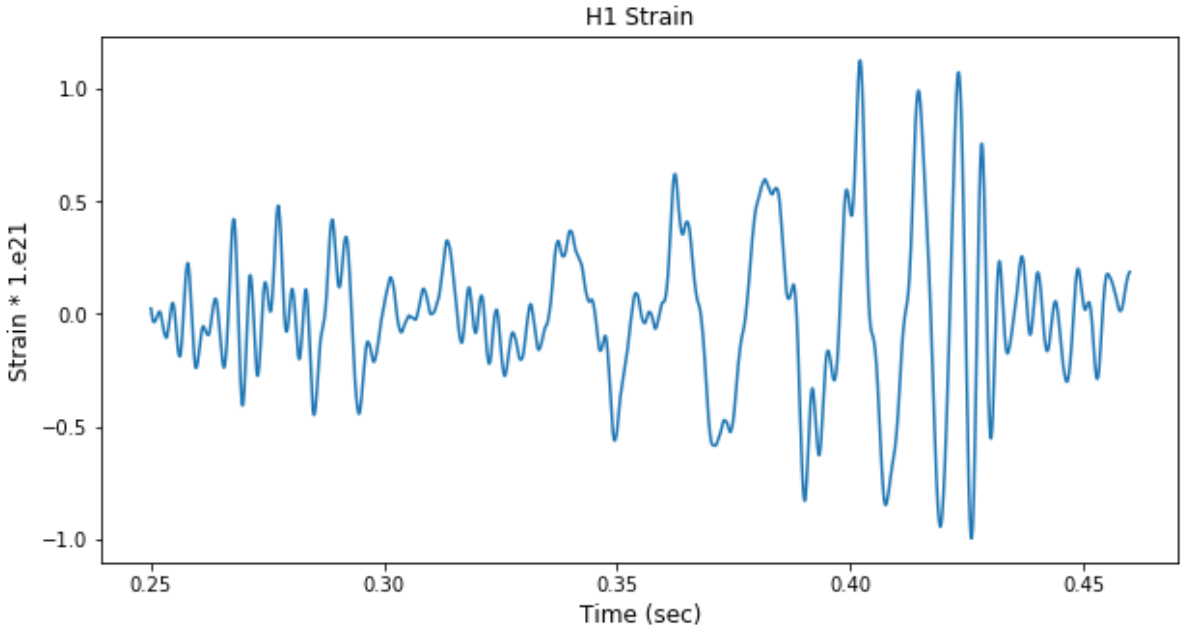
The sampling rate used was 16386. The original dataset had 3441 original points, so the sampling rate was derived from dividing the number of points by the timespan: $\frac{3441}{0.21} = 16386$.

Along with the original data and lengthened data/audio, pictured below are filtered signals. The data from LIGO is already quite clean, so filtering just a little bit has no noticeable effect, but filtering significantly does provide a smoother graph, albeit less accurate. Additionally, pictured with the audio are spectrograms, for another type of visualization of the gravitaional waves as audio.

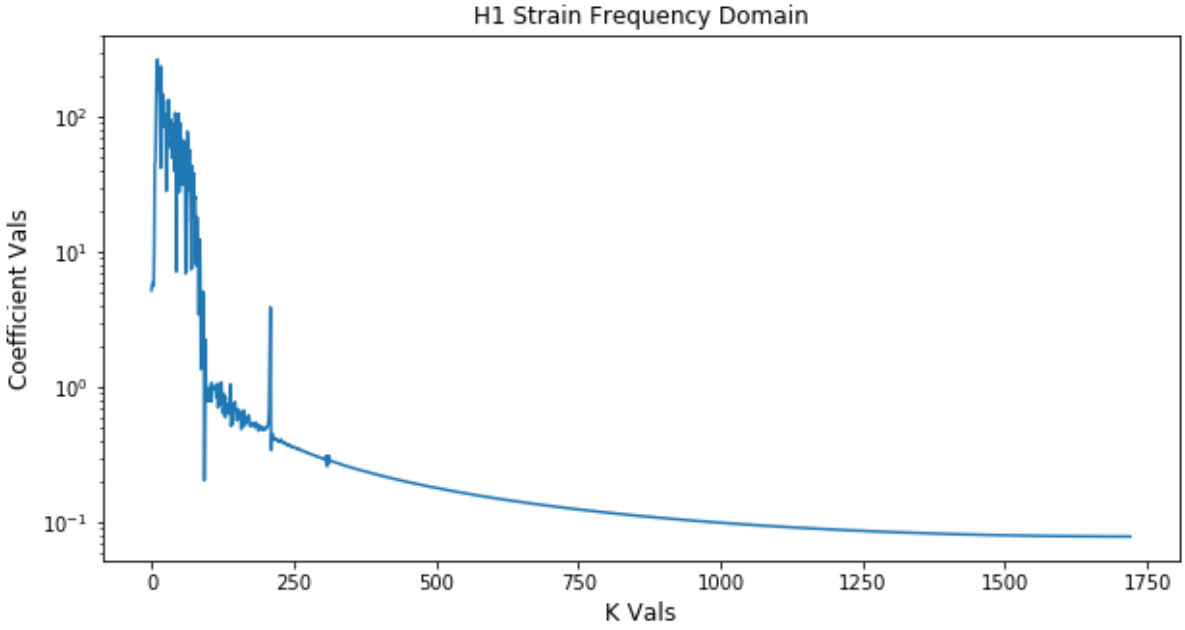
For validating what the audio sounds like, hypothetically, it is expected to sound like what the LIGO wave looks like: just some wobbling frequency.

The Hanford H1 data is shown first, and then the Livingston L1 data.

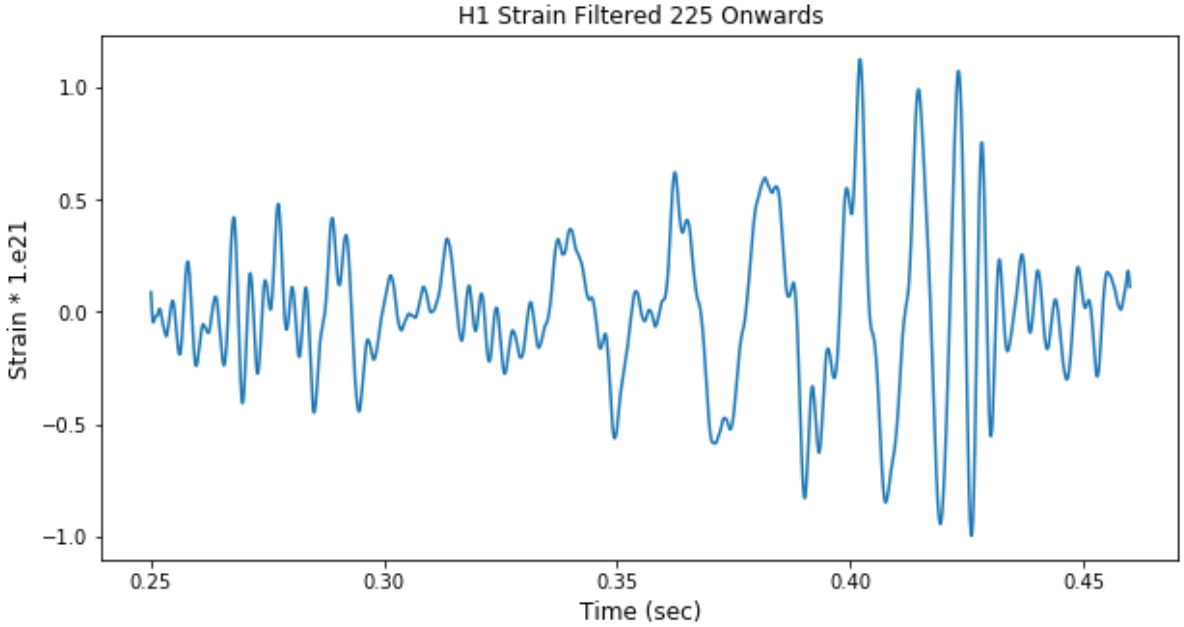
```
Out[140]: <matplotlib.text.Text at 0x11784f978>
```

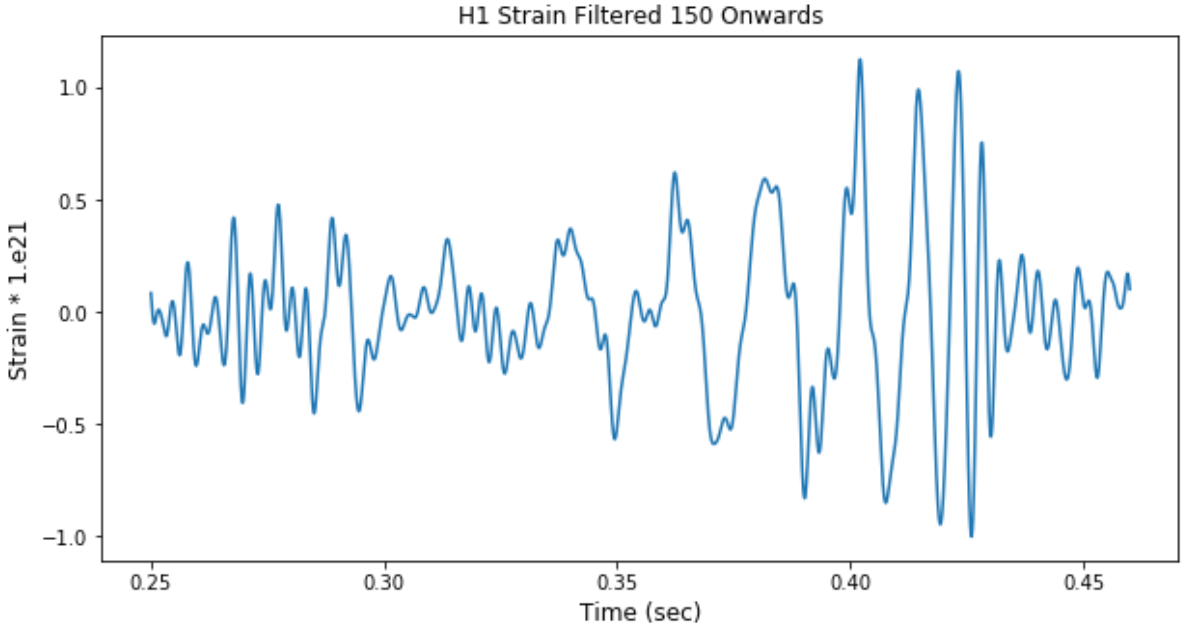


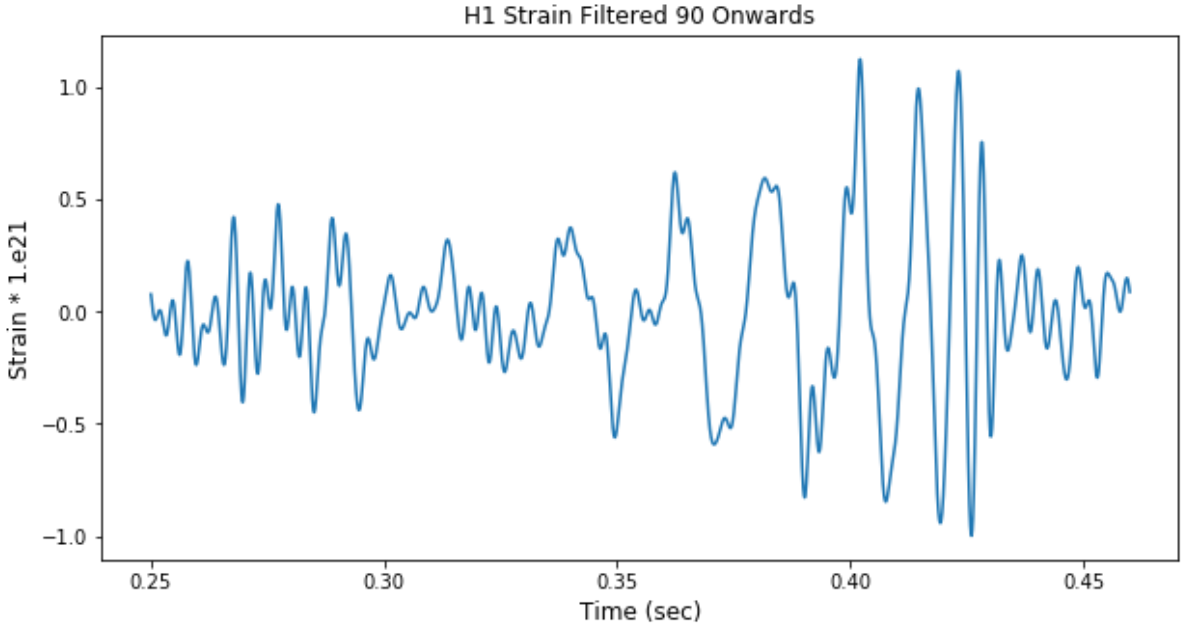

```
Out[141]: <matplotlib.text.Text at 0x129863668>
```

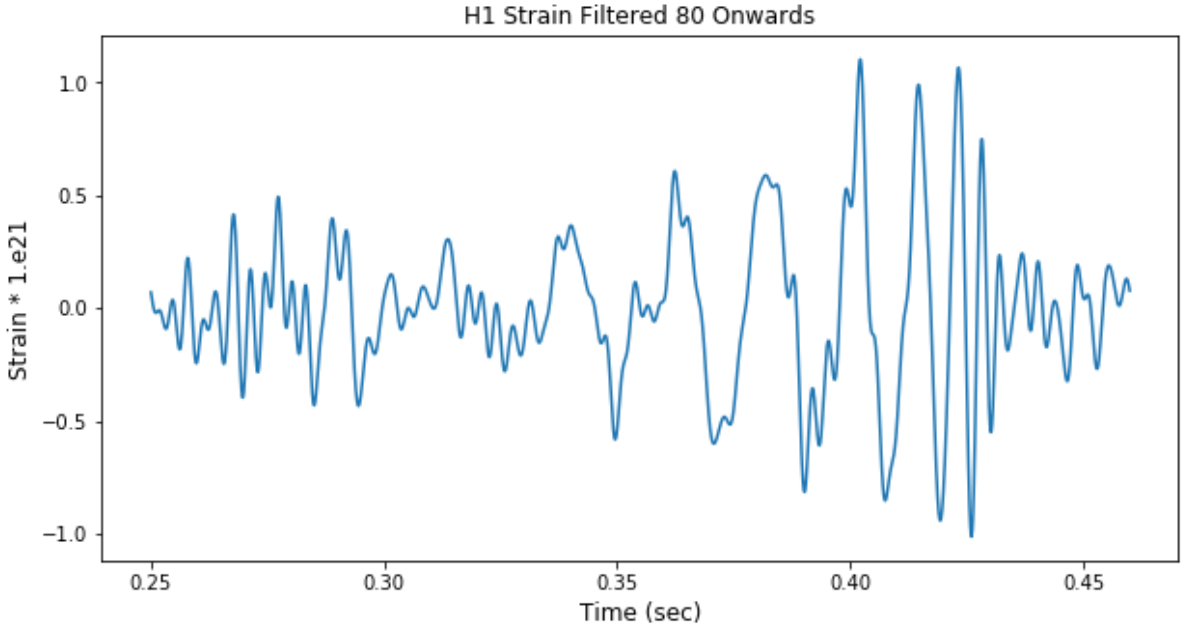


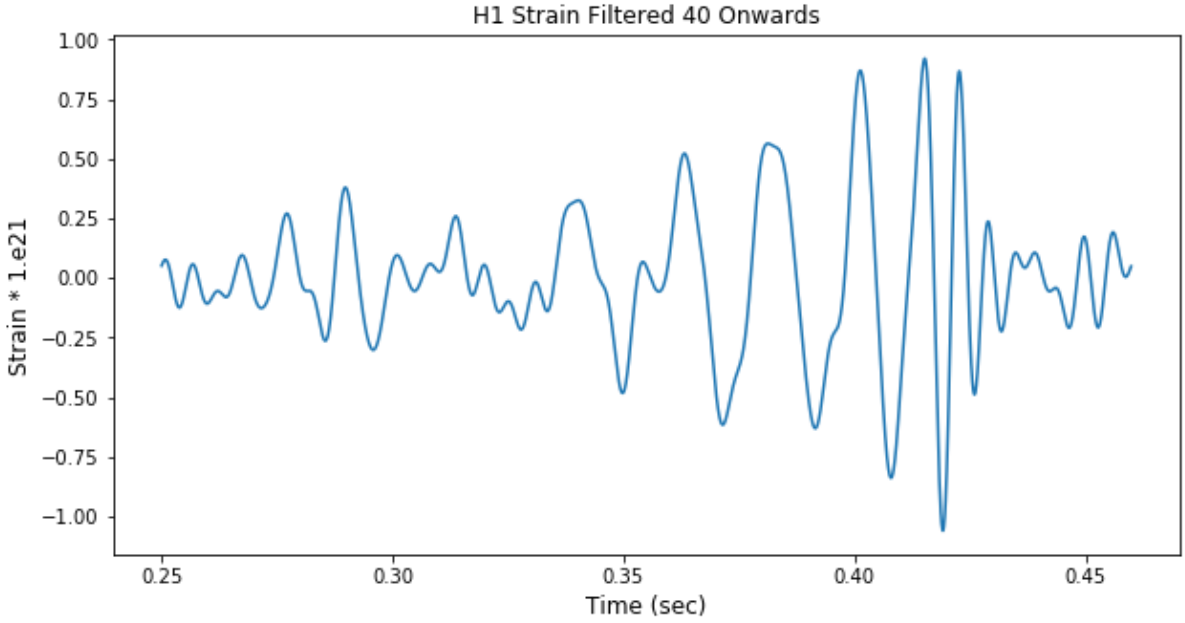
```
Out[142]: <matplotlib.text.Text at 0x10e30dcc0>
```

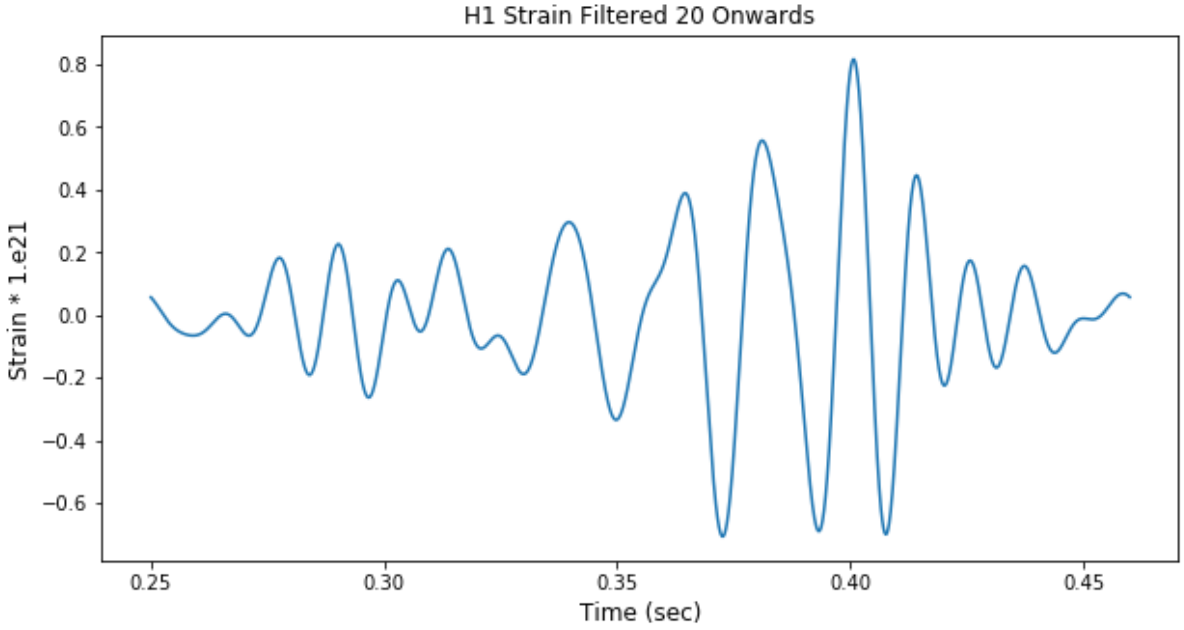

H1 Strain Filtered 225 Onwards
Strain * 1.e21
1.0
0.5
0.0
−0.5
−1.0
0.25
0.30
0.35
0.40
0.45
Time (sec)


H1 Strain Filtered 150 Onwards
Strain * 1.e21
1.0
0.5
0.0
−0.5
−1.0
0.25
0.30
0.35
0.40
0.45
Time (sec)

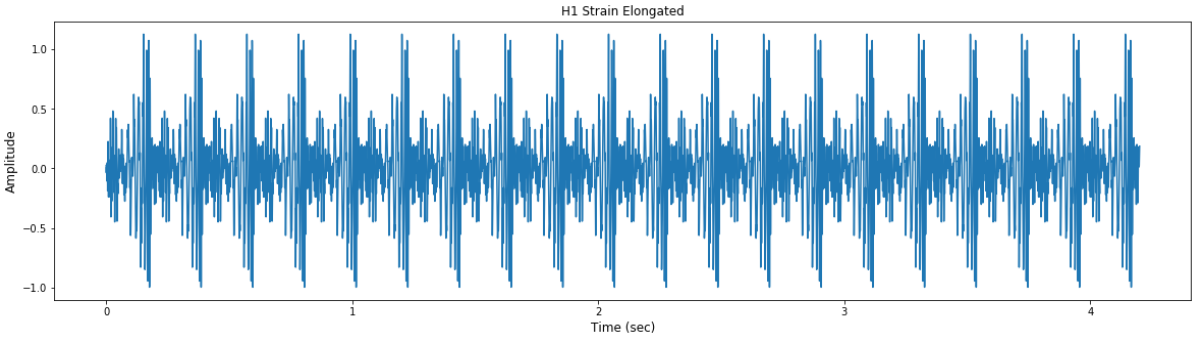

H1 Strain Filtered 90 Onwards
Strain * 1.e21
1.0
0.5
0.0
−0.5
−1.0
0.25
0.30
0.35
0.40
0.45
Time (sec)

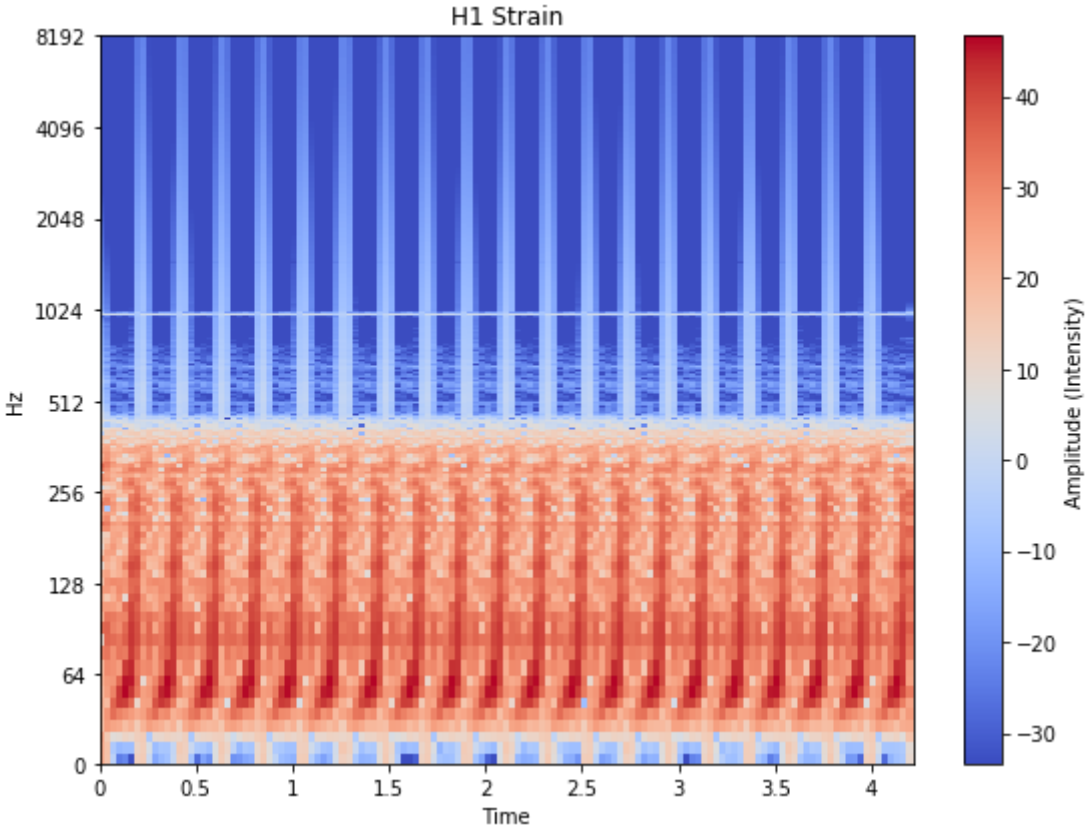

H1 Strain Filtered 80 Onwards
Strain * 1.e21
1.0
0.5
0.0
−0.5
−1.0
0.25
0.30
0.35
0.40
0.45
Time (sec)


H1 Strain Filtered 40 Onwards
Strain * 1.e21
1.00
0.75
0.50
0.25
0.00
−0.25
−0.50
−0.75
−1.00
0.25
0.30
0.35
0.40
0.45
Time (sec)

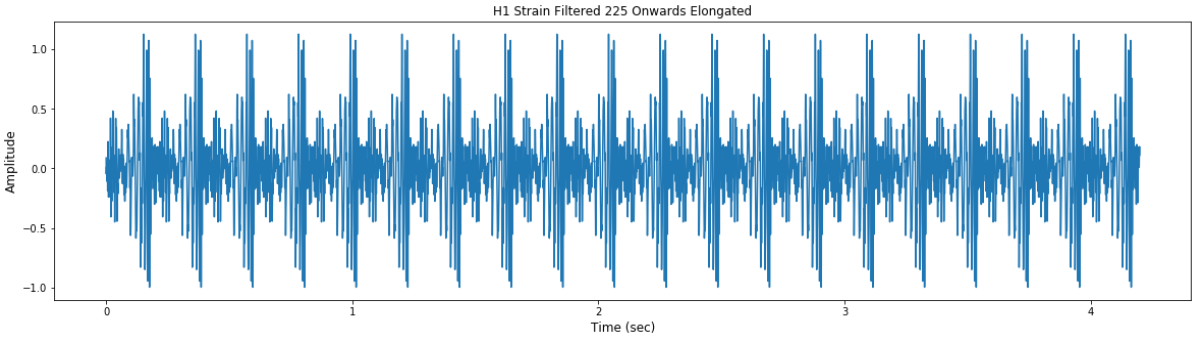

H1 Strain Filtered 20 Onwards
Strain * 1.e21
0.8
0.6
0.4
0.2
0.0
−0.2
−0.4
−0.6
0.25
0.30
0.35
0.40
0.45
Time (sec)

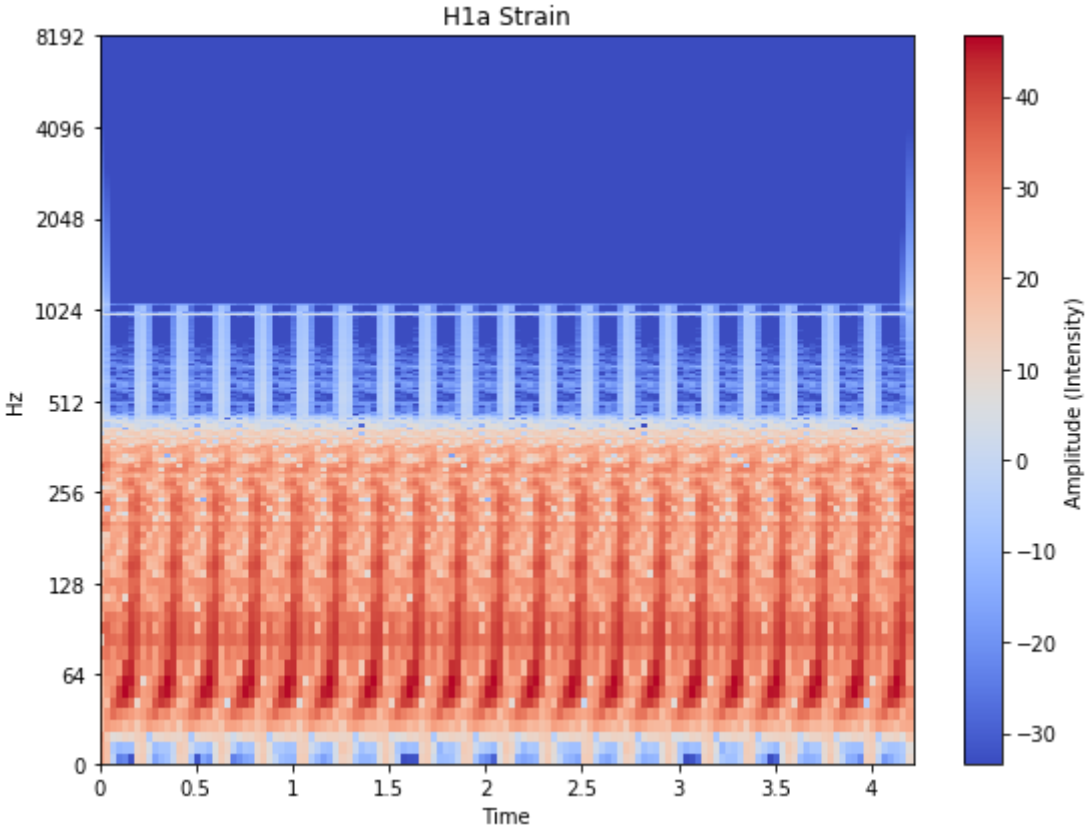

0:00 / 0:04


H1 Strain Elongated
Amplitude
Time (sec)

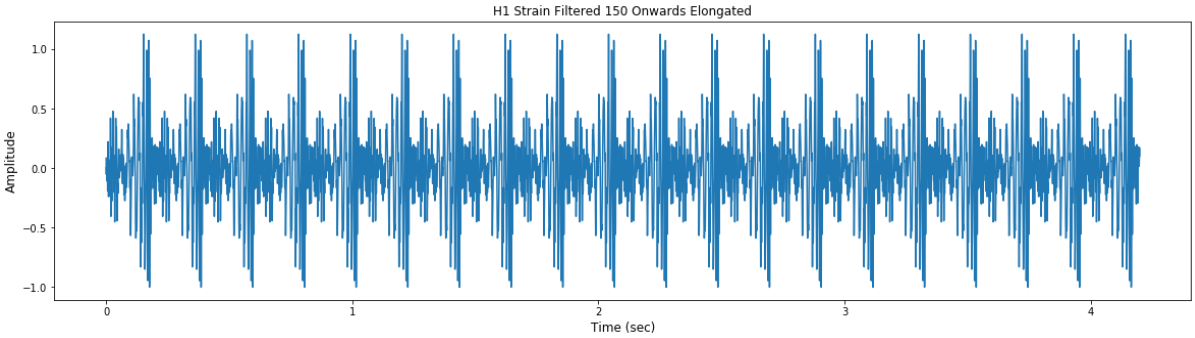

H1 Strain
Hz
Time
Amplitude (Intensity)

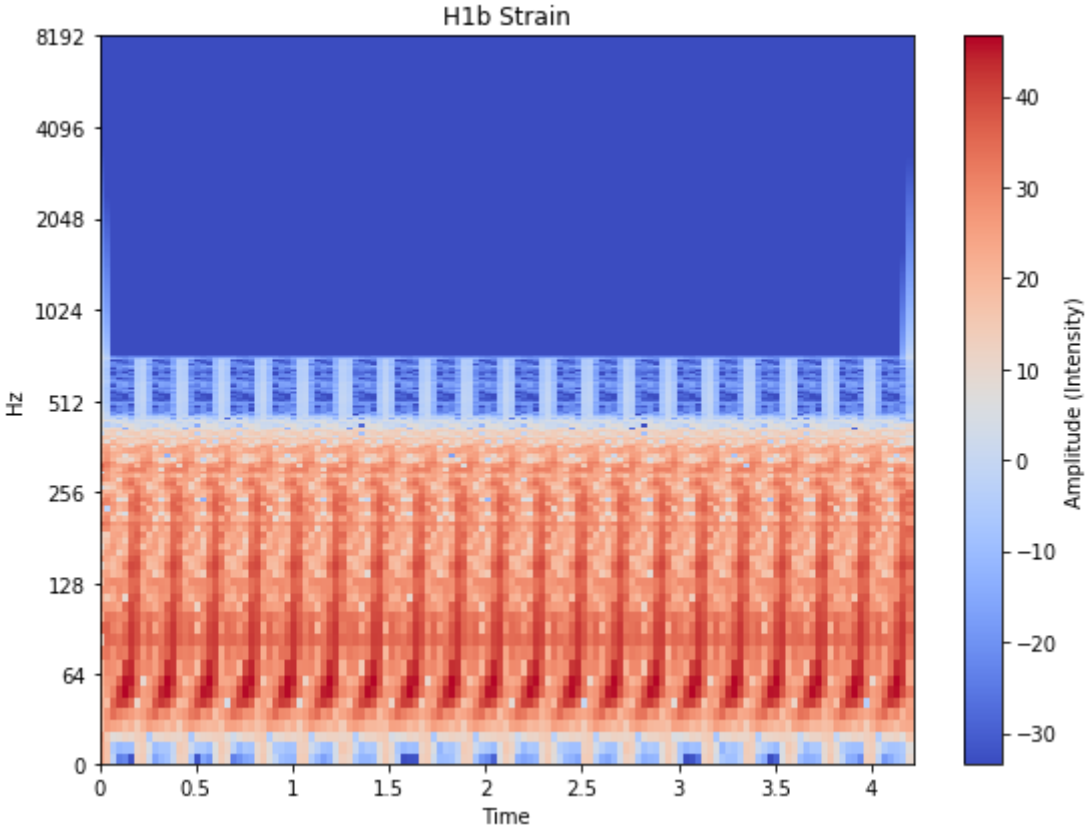

0:00 / 0:04


H1 Strain Filtered 225 Onwards Elongated
Amplitude
Time (sec)

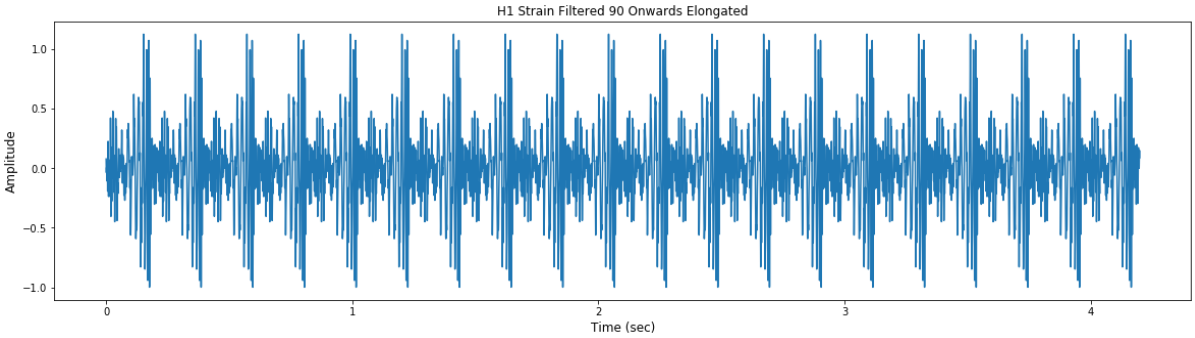

H1a Strain
Hz
Time
Amplitude (Intensity)

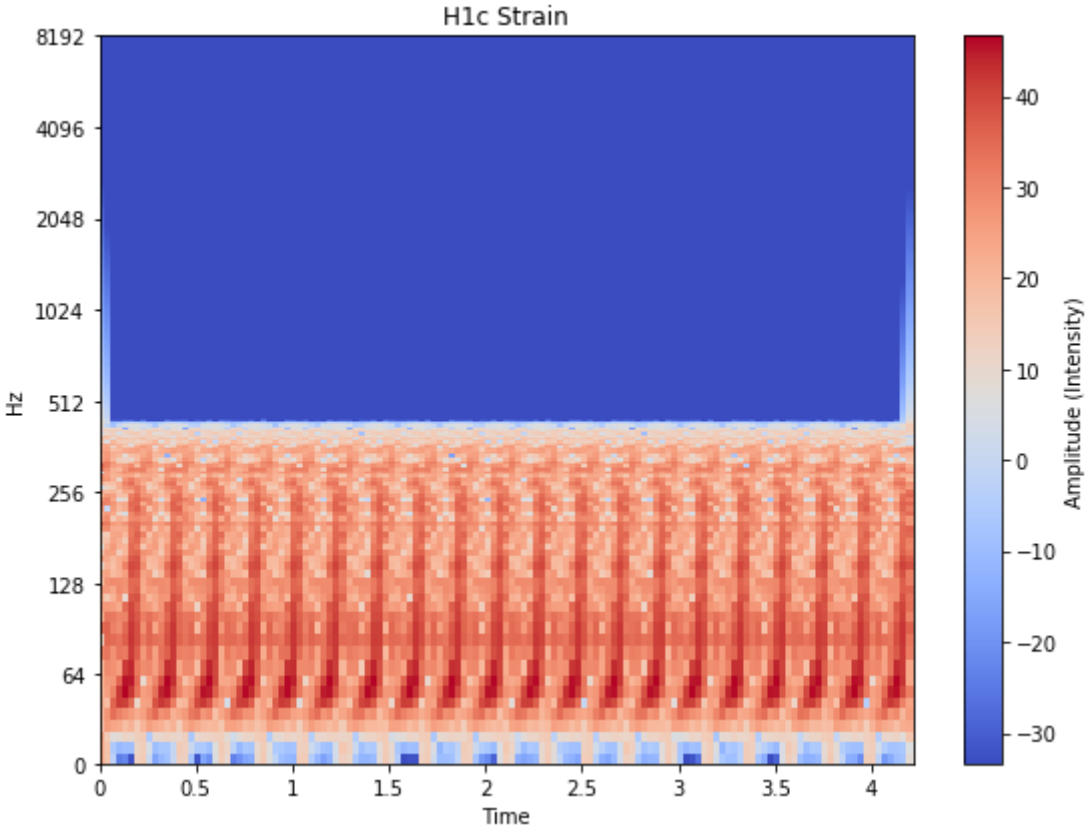

0:00 / 0:04


H1 Strain Filtered 150 Onwards Elongated
Amplitude
Time (sec)

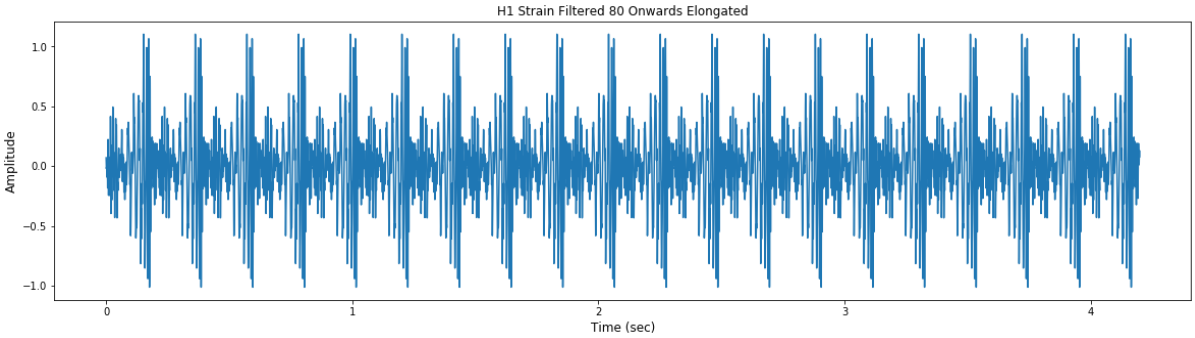

H1b Strain
Hz
Time
Amplitude (Intensity)

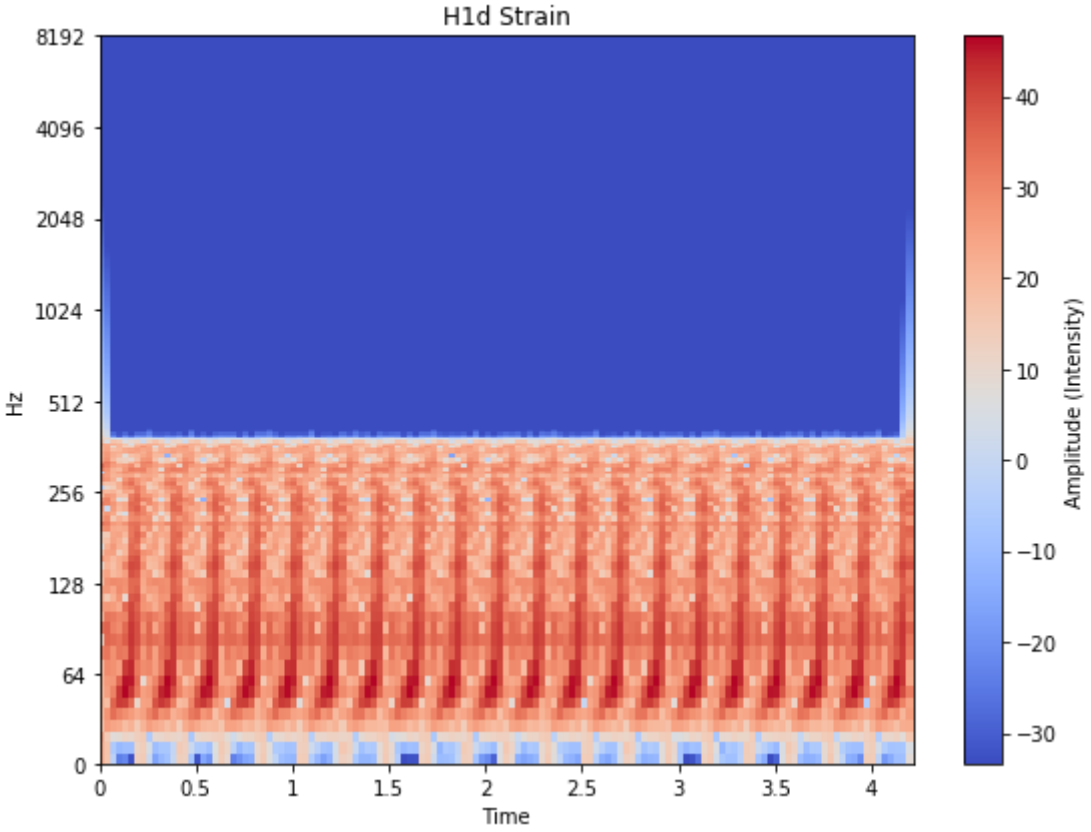

0:00 / 0:04


H1 Strain Filtered 90 Onwards Elongated
Amplitude
Time (sec)

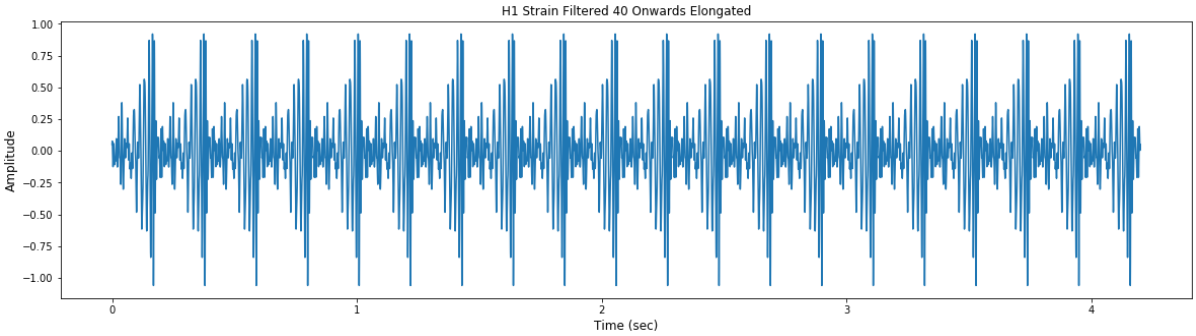

H1c Strain
Hz
Time
Amplitude (Intensity)

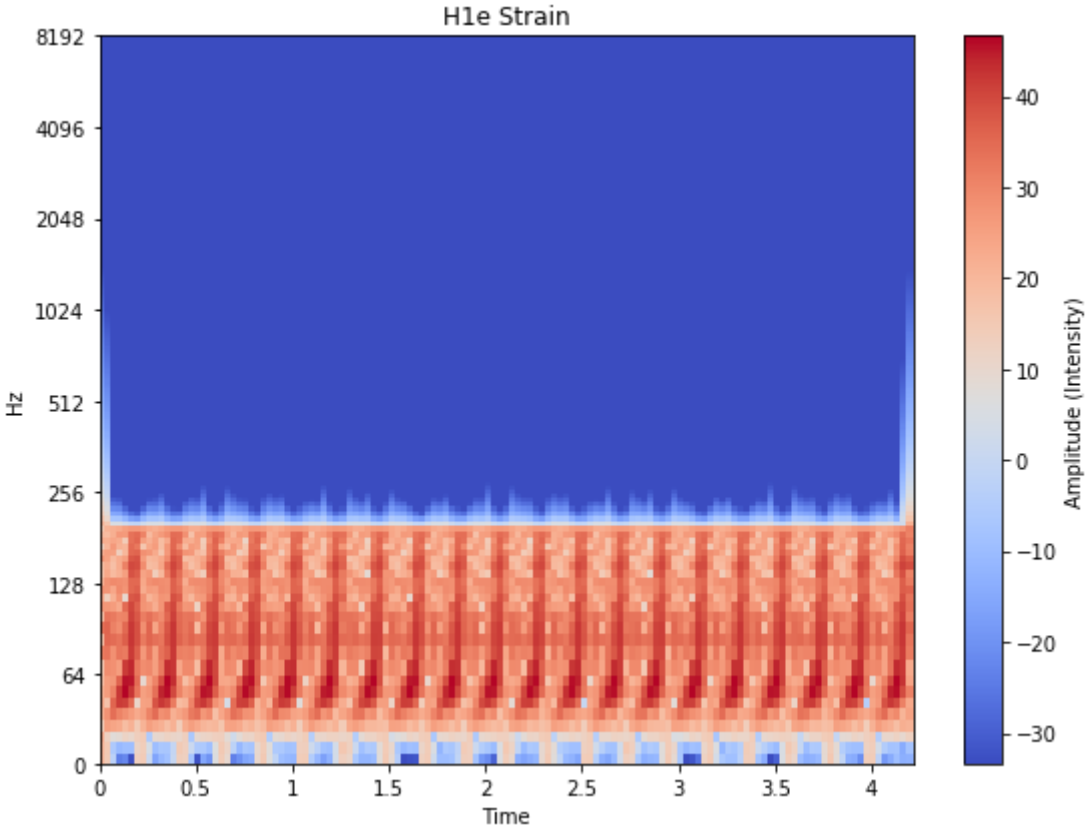

0:00 / 0:04


H1 Strain Filtered 80 Onwards Elongated
Amplitude
Time (sec)

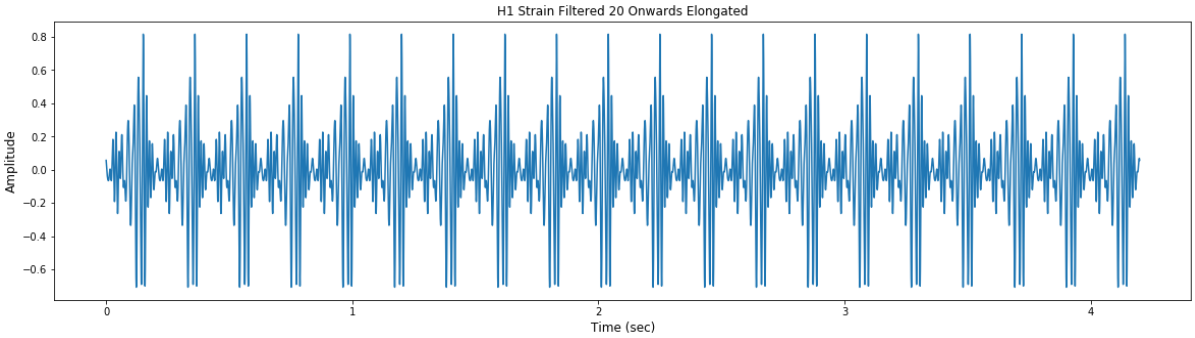

H1d Strain
Hz
Time
Amplitude (Intensity)

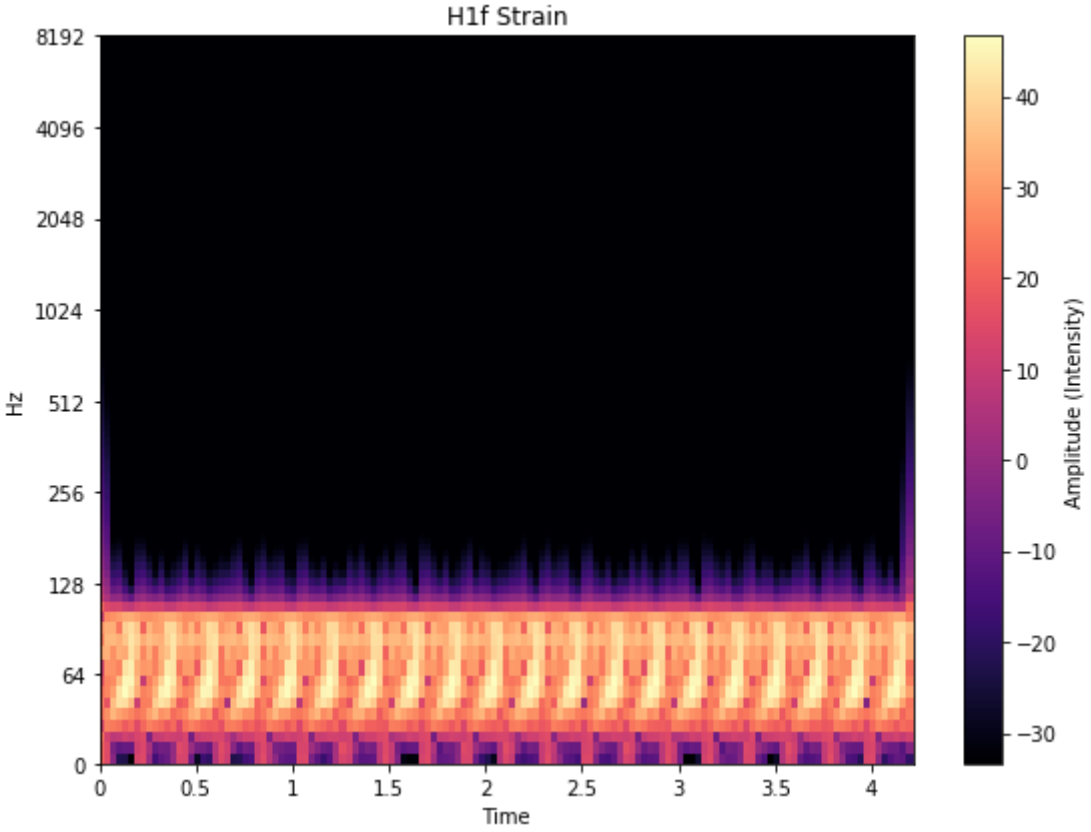

0:00 / 0:04


H1 Strain Filtered 40 Onwards Elongated
Amplitude
Time (sec)

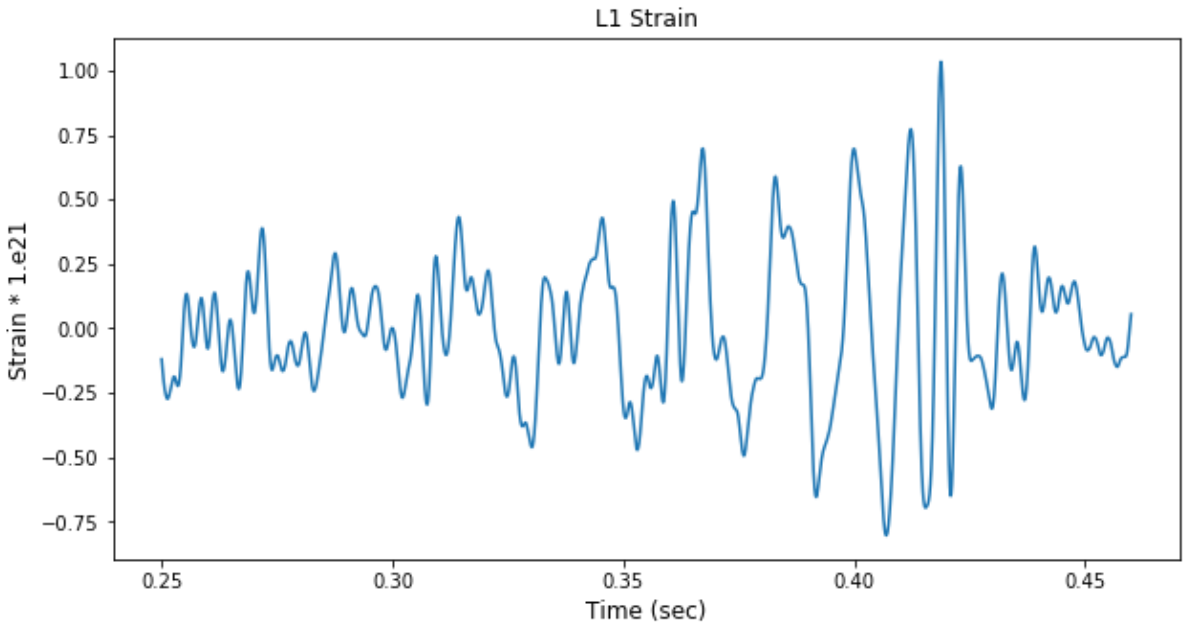

H1e Strain
Hz
Time
Amplitude (Intensity)


0:00 / 0:04

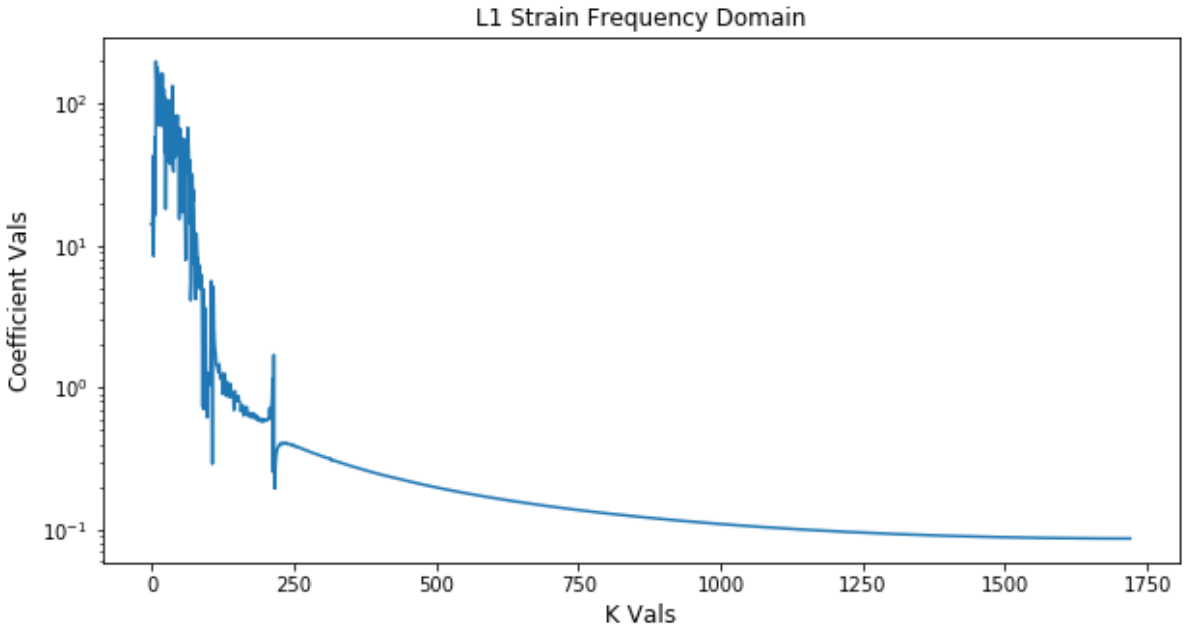

H1 Strain Filtered 20 Onwards Elongated
Amplitude
Time (sec)


H1f Strain
Hz
Time
Amplitude (Intensity)

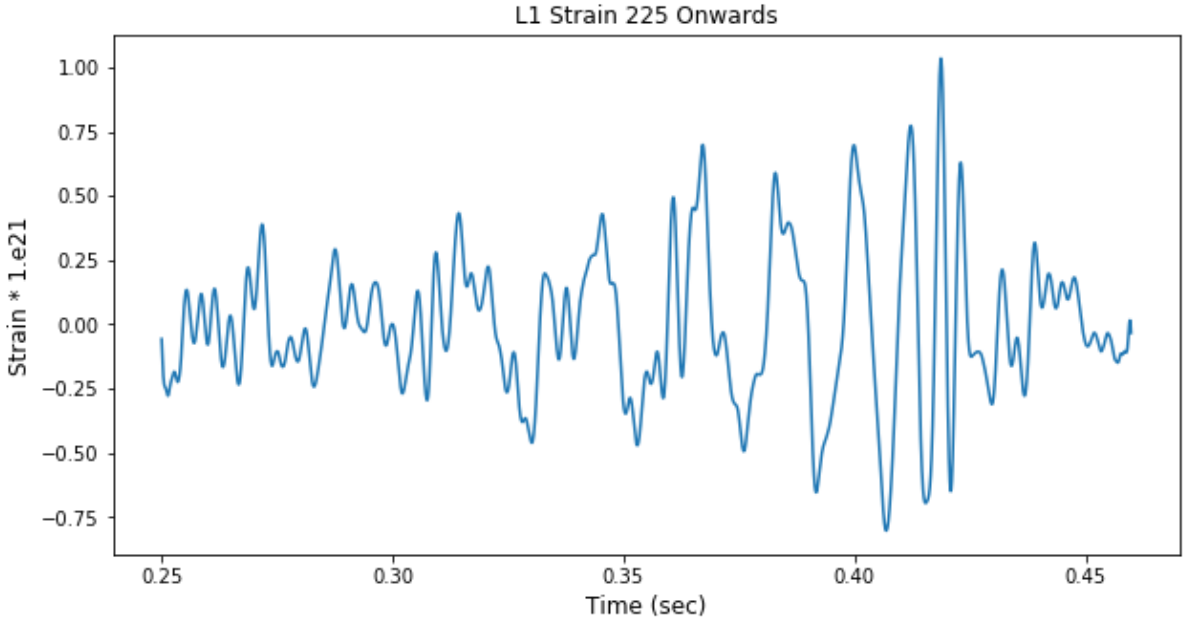
Out[183]: <matplotlib.text.Text at 0x11acba518>

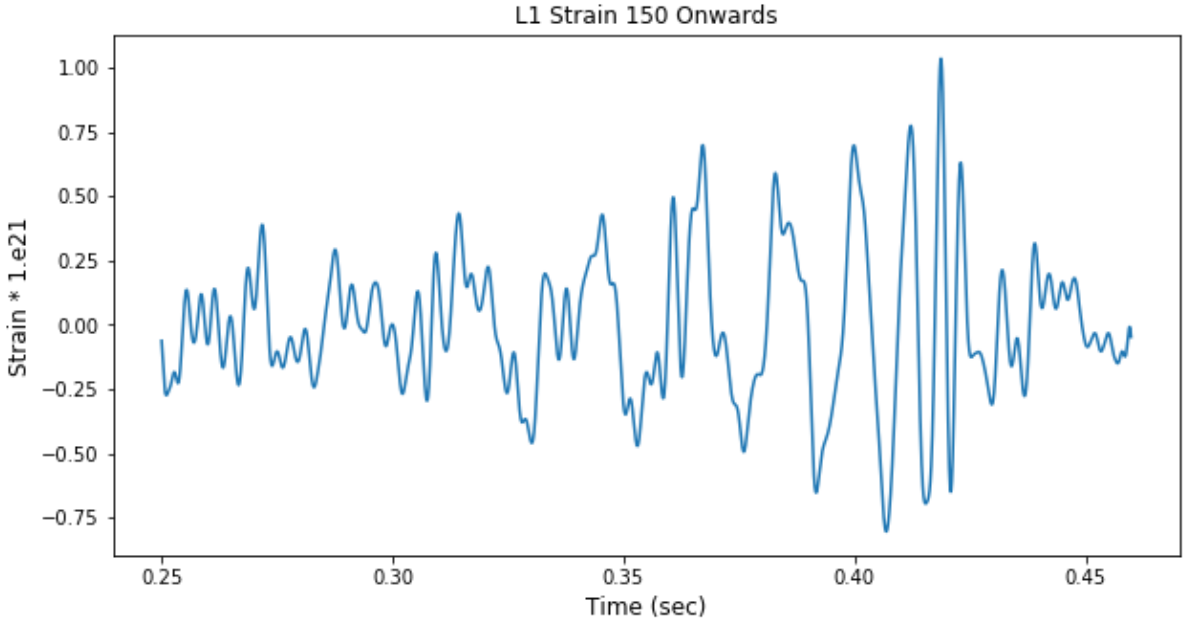


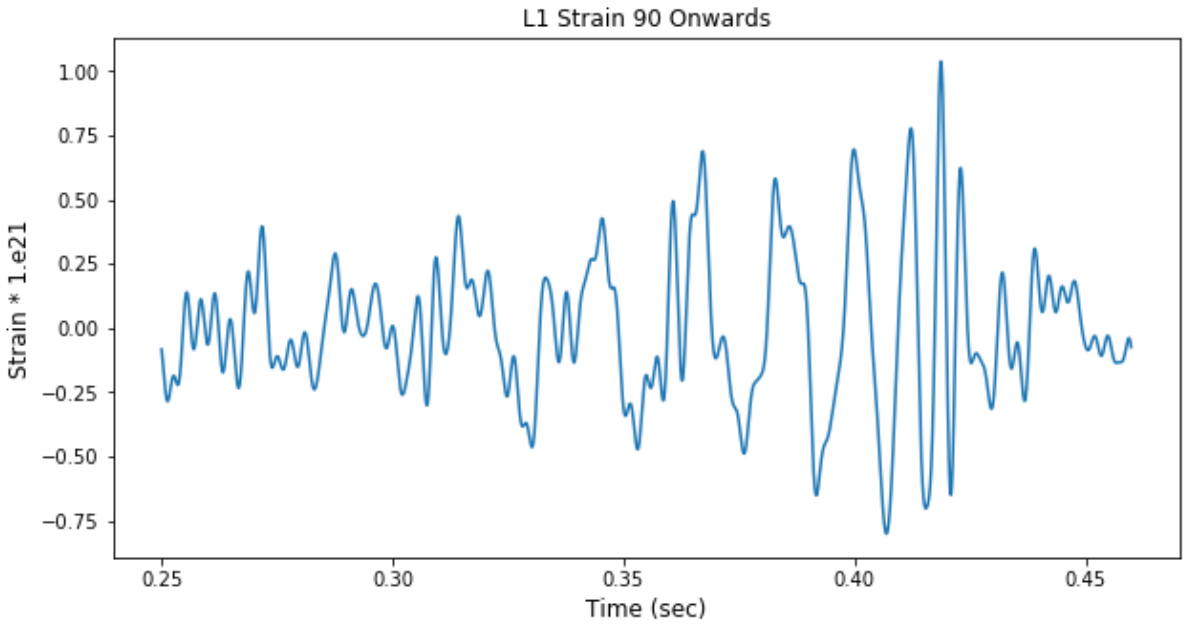
Out[184]: <matplotlib.text.Text at 0x11bbc8e10>

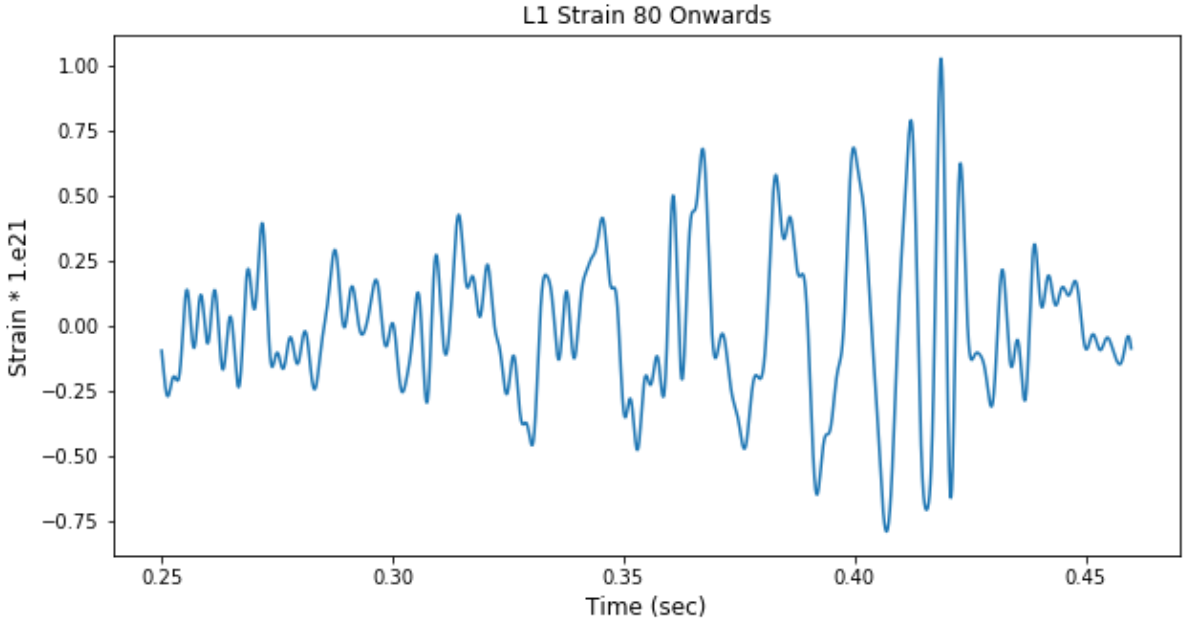


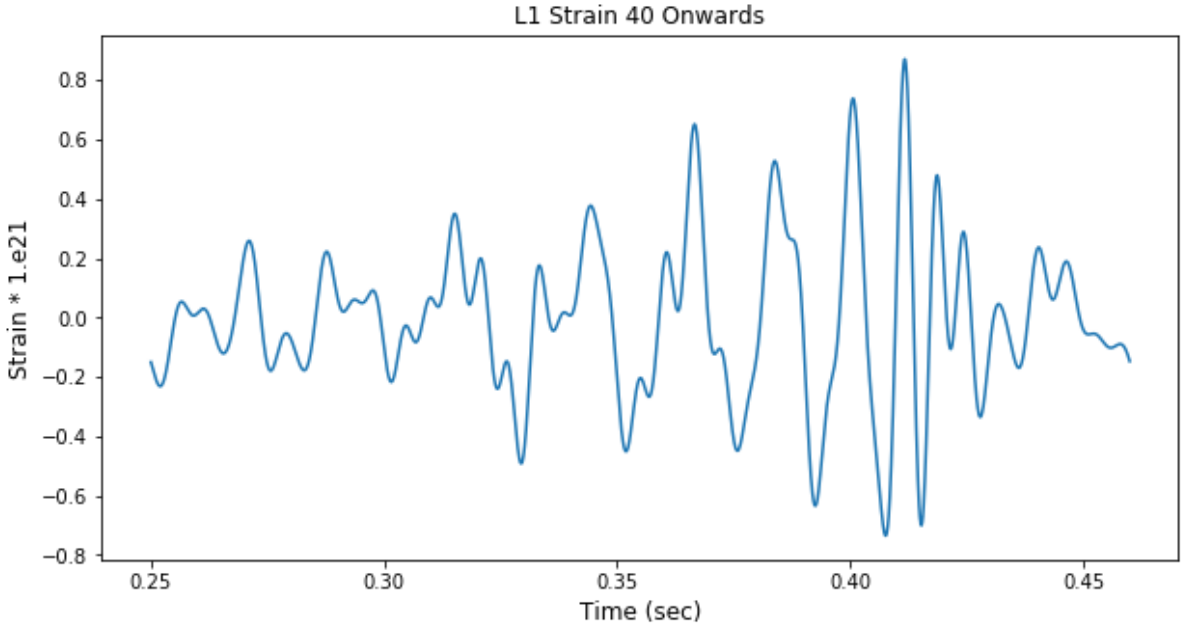
```
Out[185]: <matplotlib.text.Text at 0x11aab0e48>
```

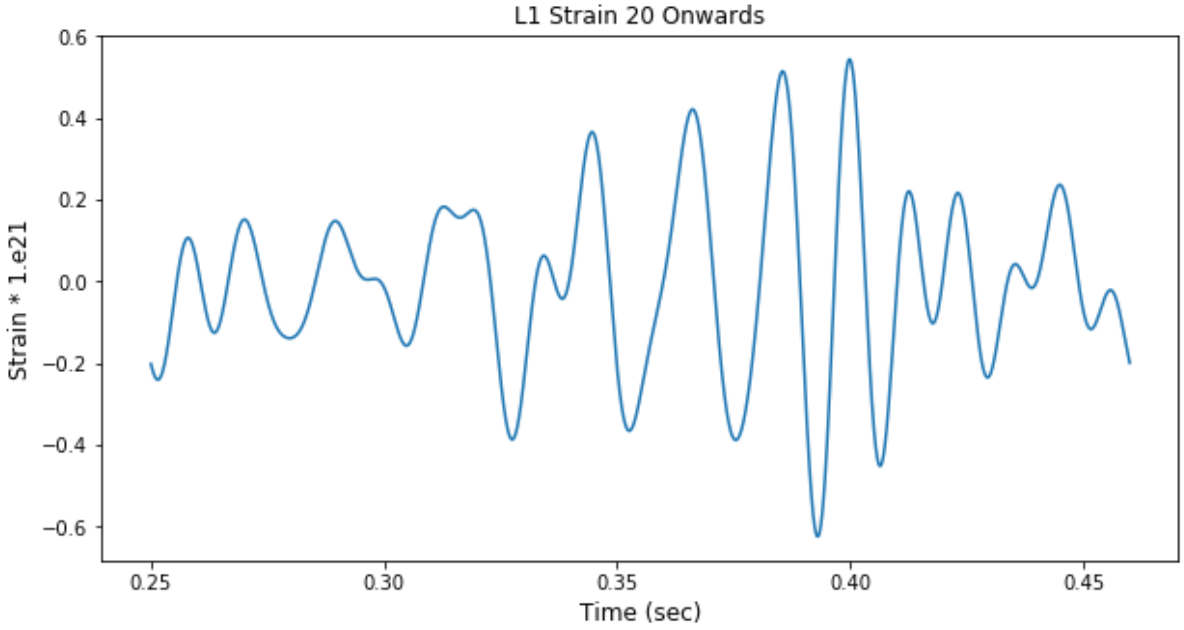

L1 Strain 225 Onwards
Strain * 1.e21
1.00
0.75
0.50
0.25
0.00
−0.25
−0.50
−0.75
0.25
0.30
0.35
0.40
0.45
Time (sec)


L1 Strain 150 Onwards
Strain * 1.e21
1.00
0.75
0.50
0.25
0.00
−0.25
−0.50
−0.75
0.25
0.30
0.35
0.40
0.45
Time (sec)

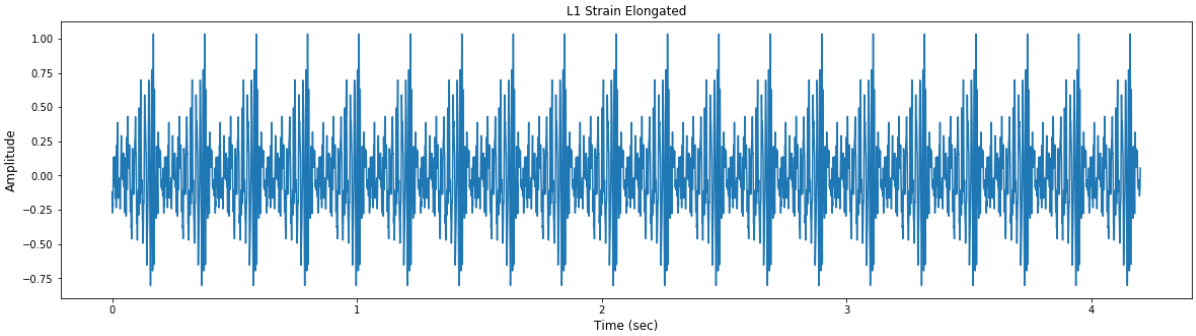

L1 Strain 90 Onwards
Strain * 1.e21
1.00
0.75
0.50
0.25
0.00
−0.25
−0.50
−0.75
0.25
0.30
0.35
0.40
0.45
Time (sec)

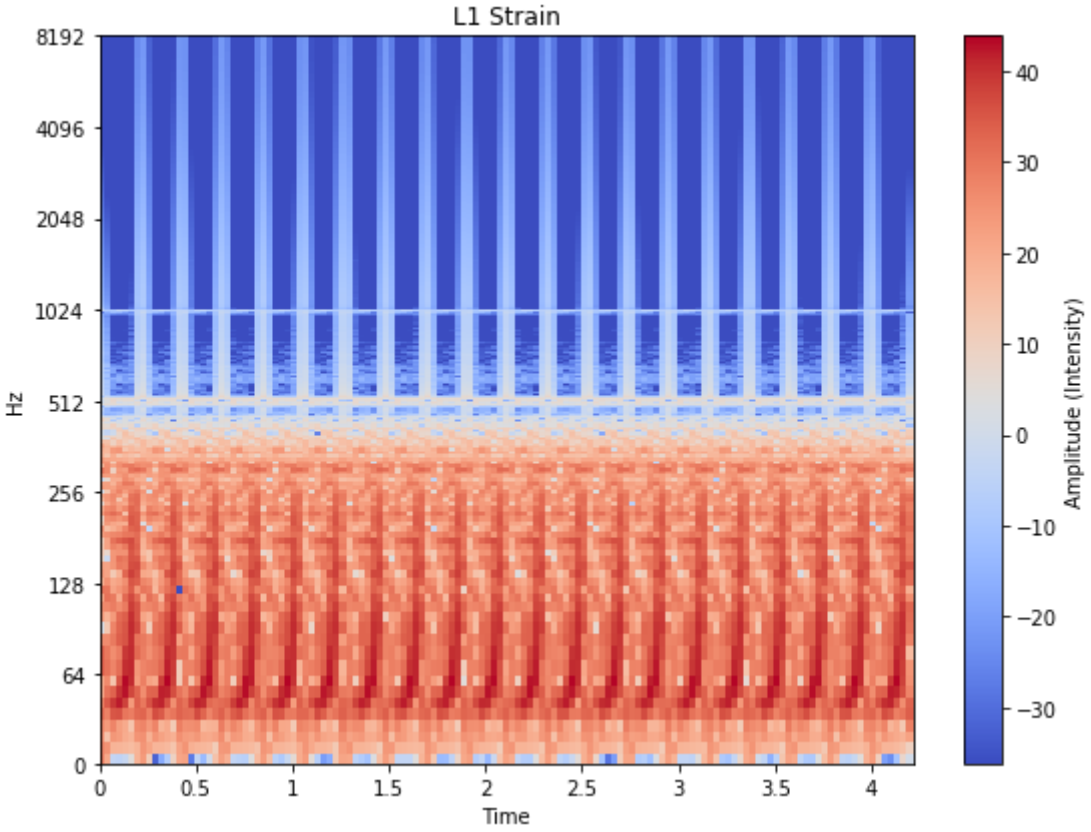

L1 Strain 80 Onwards
Strain * 1.e21
1.00
0.75
0.50
0.25
0.00
−0.25
−0.50
−0.75
0.25
0.30
0.35
0.40
0.45
Time (sec)


L1 Strain 40 Onwards
Strain * 1.e21
0.8
0.6
0.4
0.2
0.0
−0.2
−0.4
−0.6
−0.8
0.25
0.30
0.35
0.40
0.45
Time (sec)

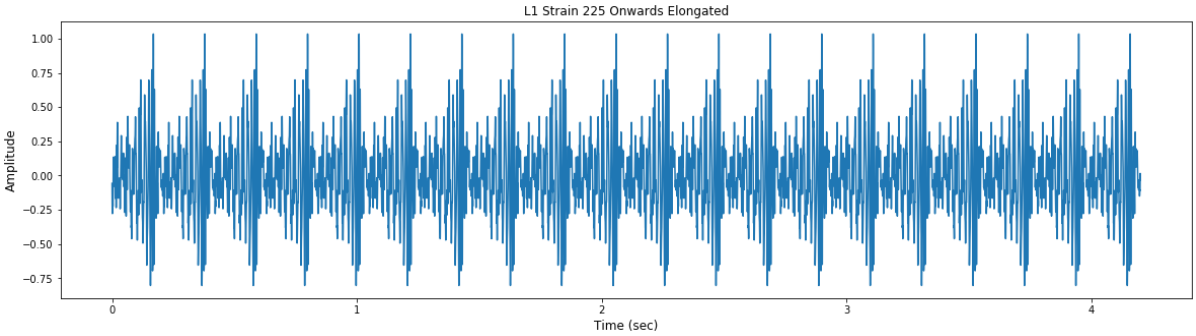

L1 Strain 20 Onwards
Strain * 1.e21
0.6
0.4
0.2
0.0
−0.2
−0.4
−0.6
0.25
0.30
0.35
0.40
0.45
Time (sec)

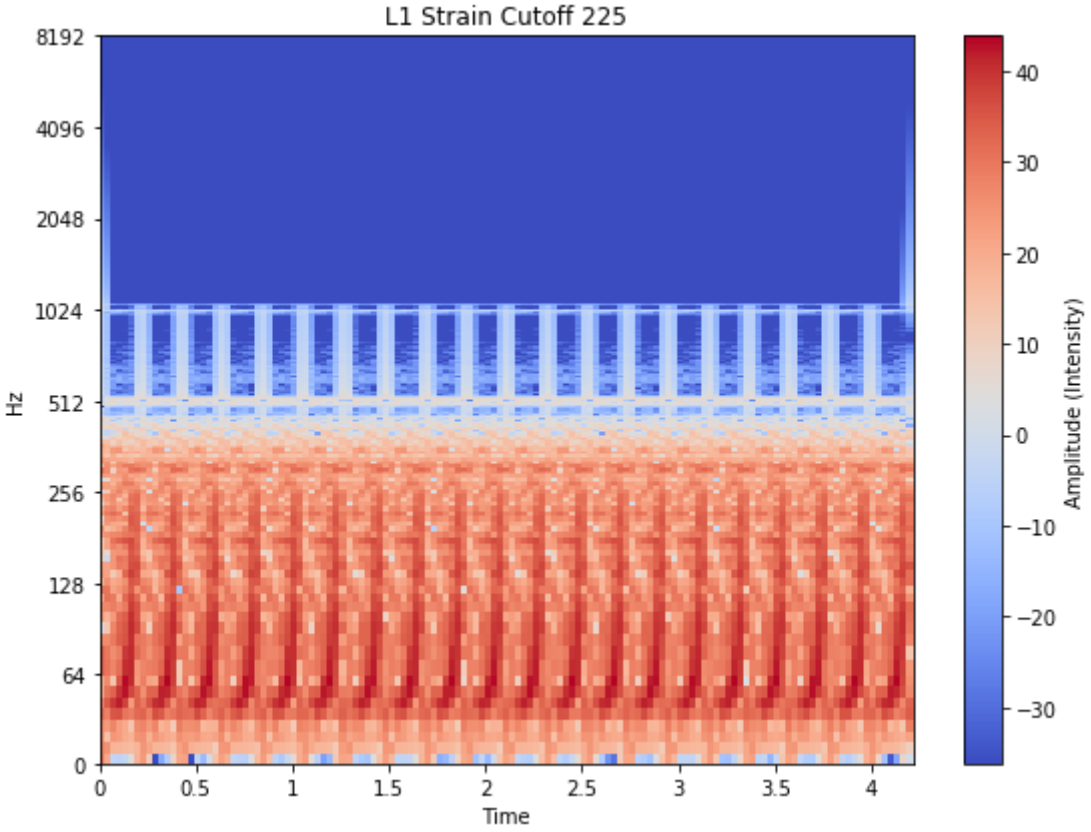

0:00 / 0:04


L1 Strain Elongated
Amplitude
Time (sec)

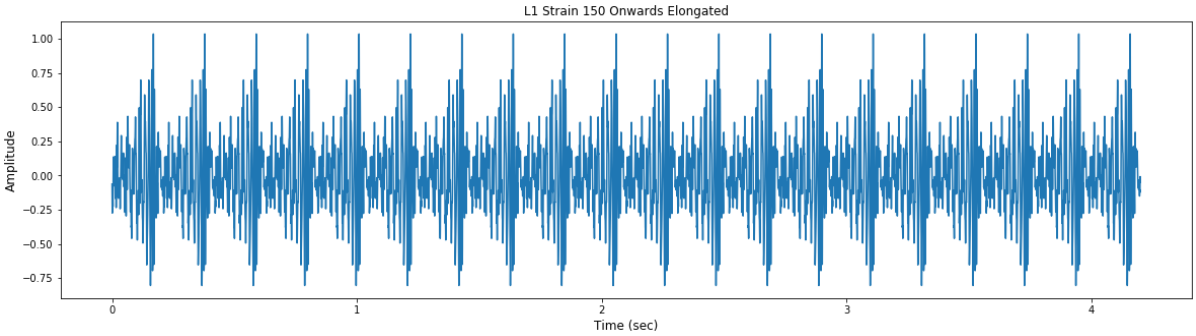

L1 Strain
Hz
Time
Amplitude (Intensity)

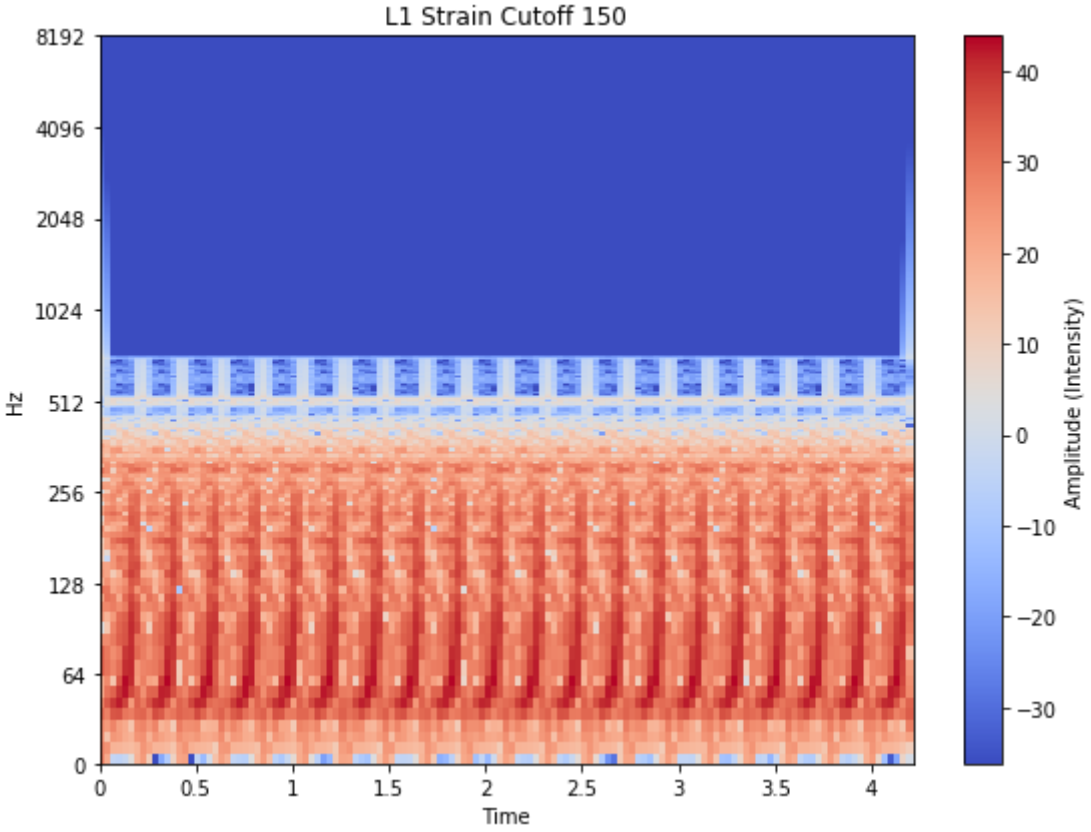

0:00 / 0:04


L1 Strain 225 Onwards Elongated
Amplitude
Time (sec)

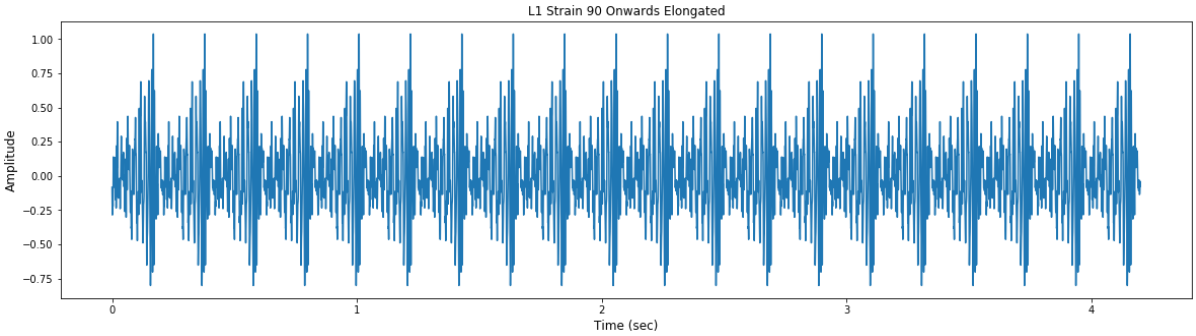

L1 Strain Cutoff 225
Hz
Time
Amplitude (Intensity)

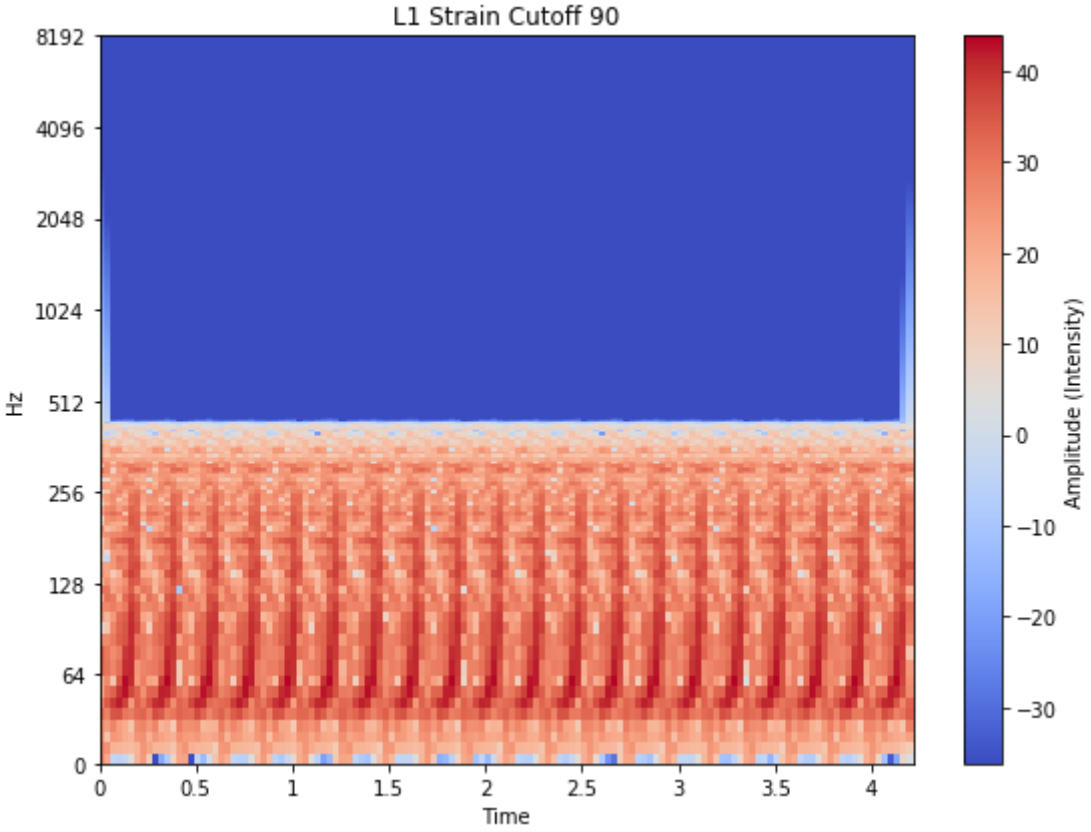

0:00 / 0:04


L1 Strain 150 Onwards Elongated
Amplitude
Time (sec)

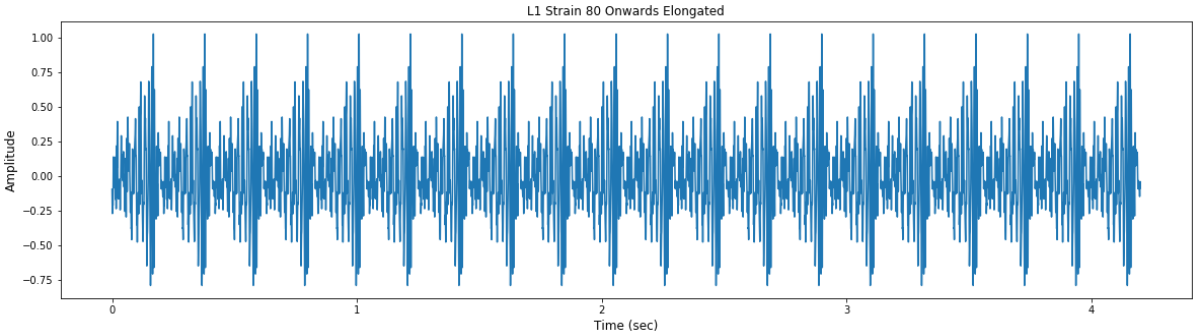

L1 Strain Cutoff 150
Hz
Time
Amplitude (Intensity)

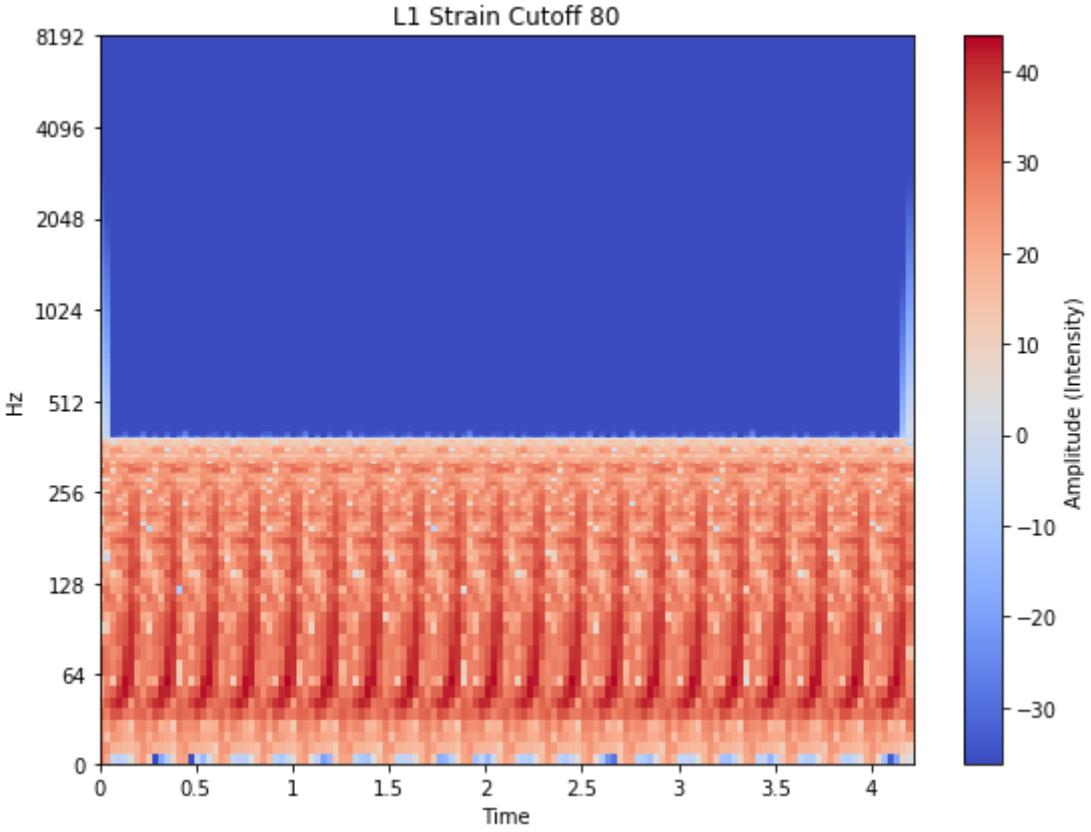

0:00 / 0:04


L1 Strain 90 Onwards Elongated
Amplitude
Time (sec)

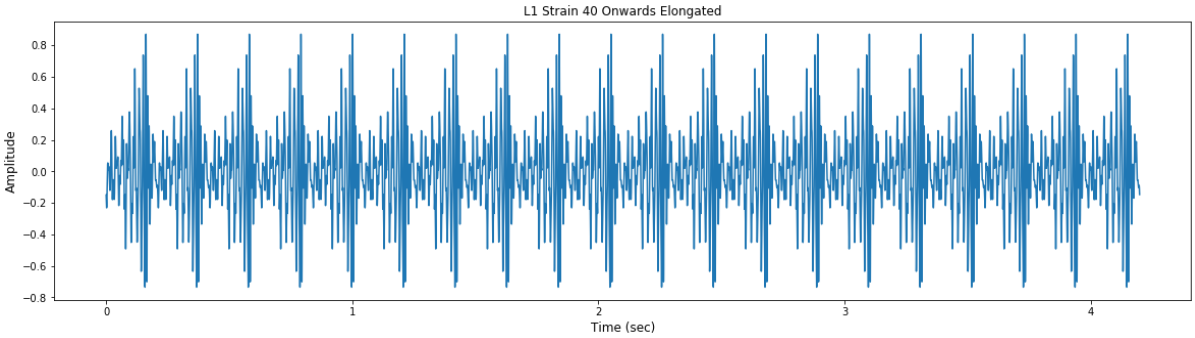

L1 Strain Cutoff 90
Hz
Time
Amplitude (Intensity)

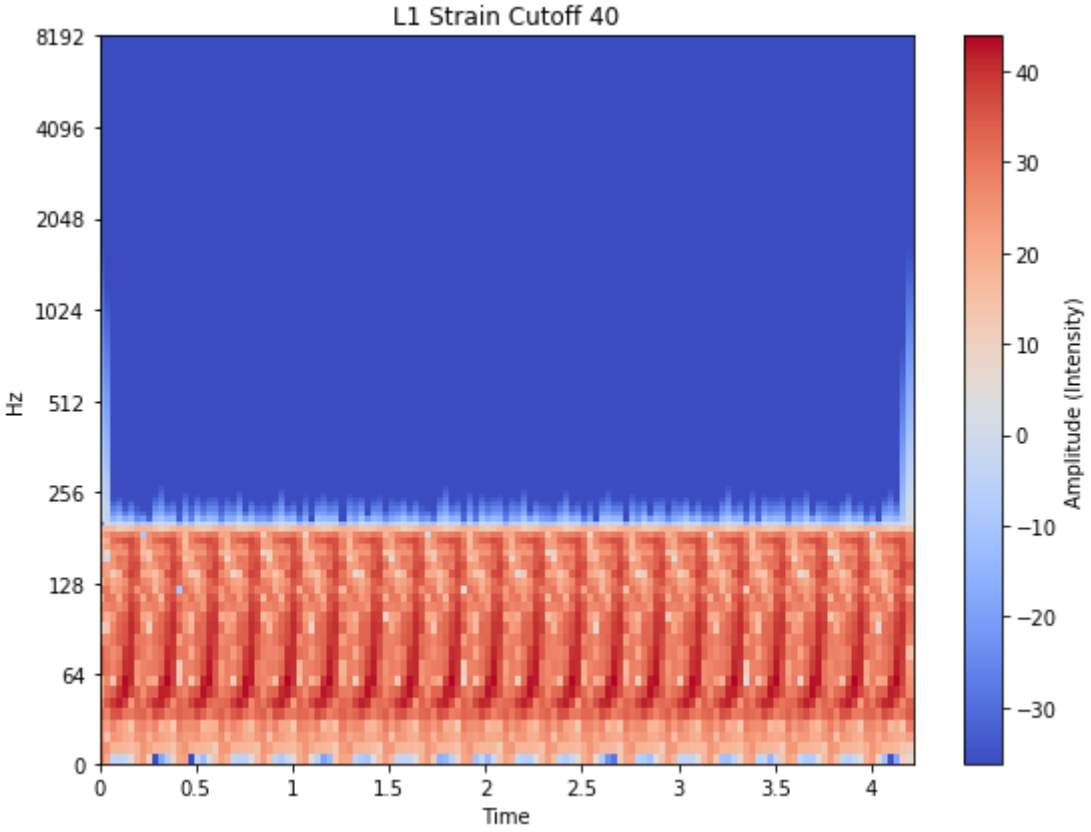

0:00 / 0:04


L1 Strain 80 Onwards Elongated
Amplitude
Time (sec)

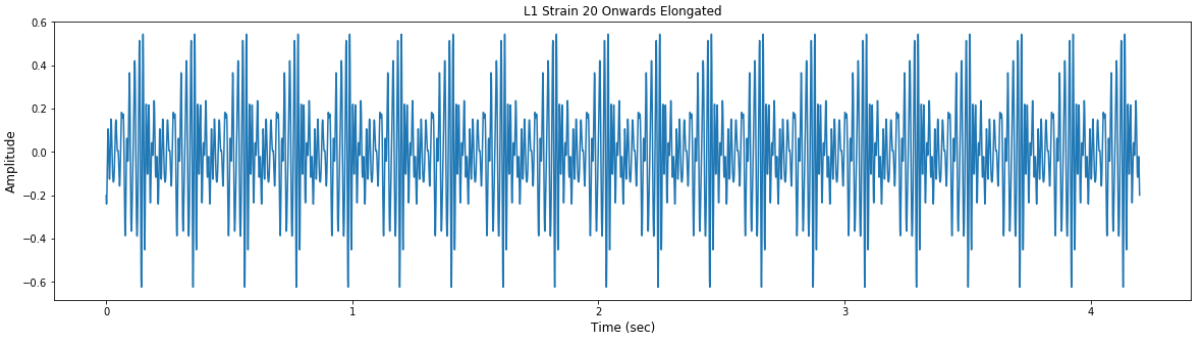

L1 Strain Cutoff 80
Hz
Time
Amplitude (Intensity)

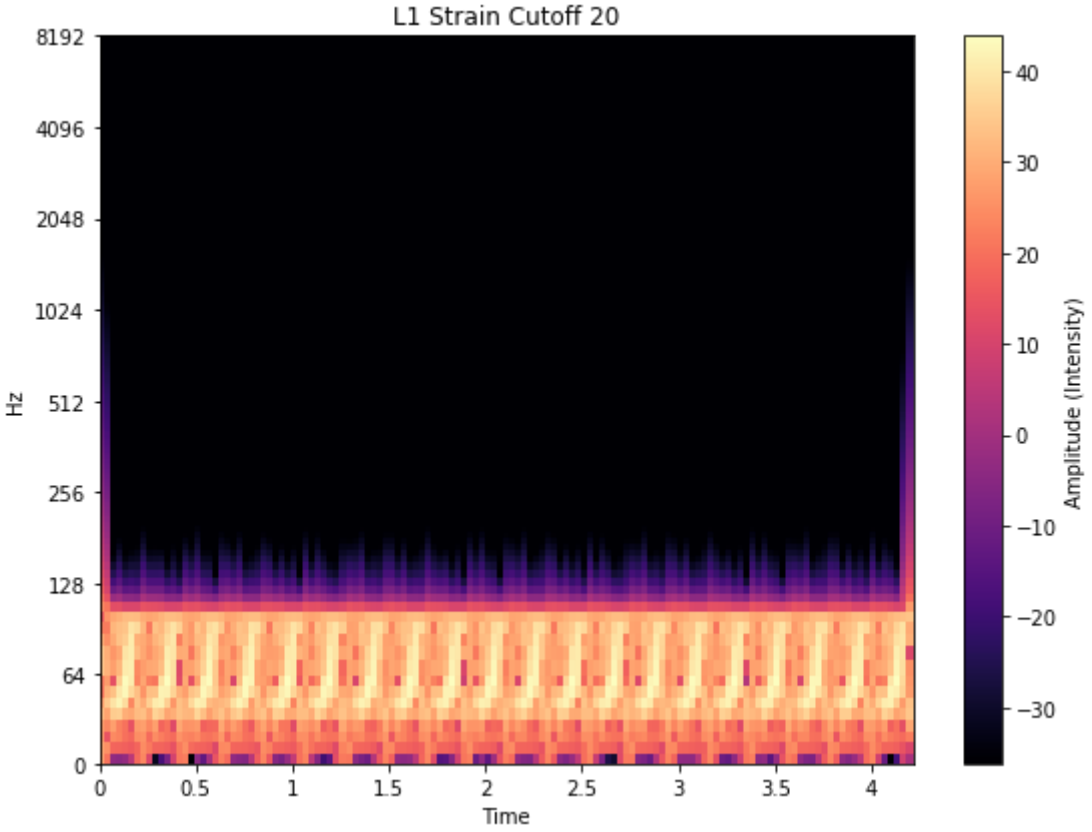

0:00 / 0:04

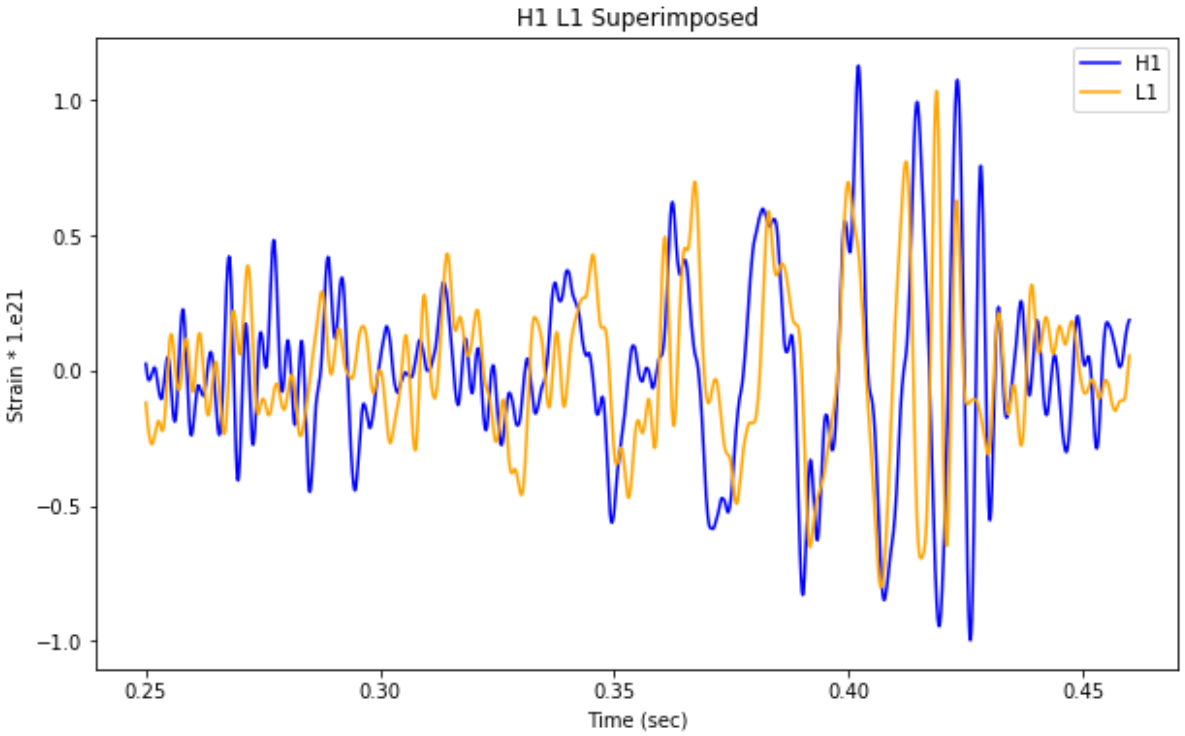

L1 Strain 40 Onwards Elongated
Amplitude
Time (sec)

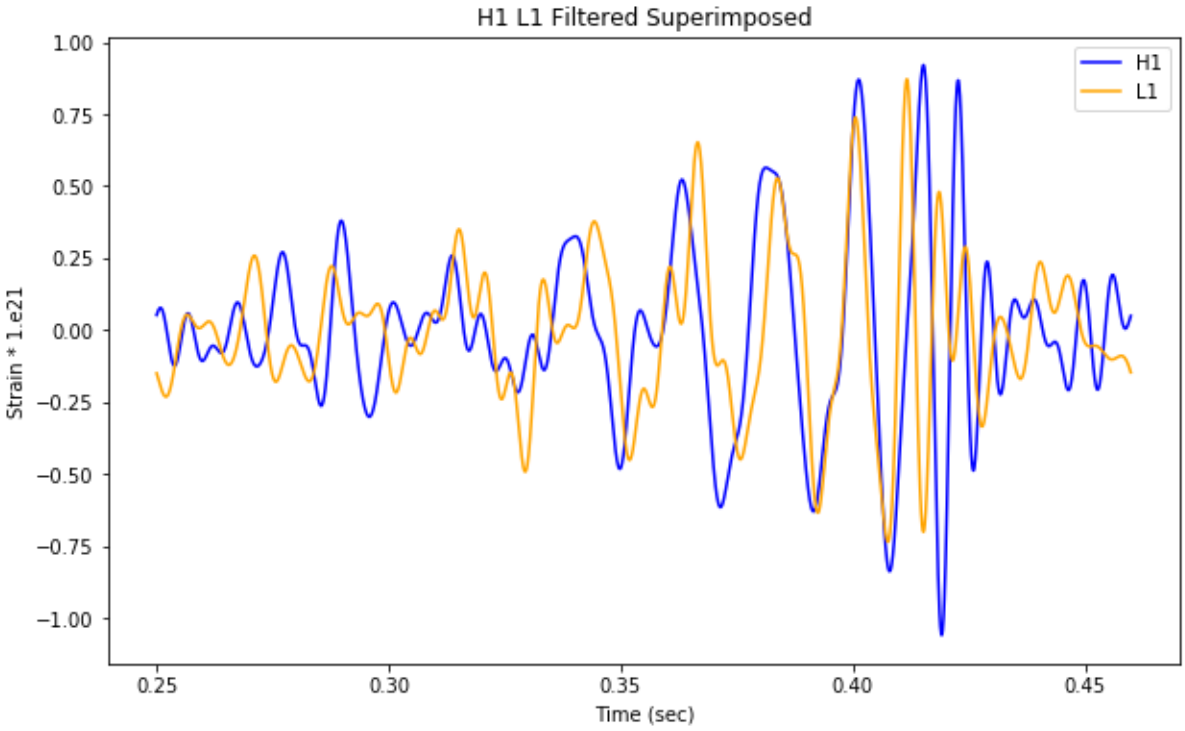

L1 Strain Cutoff 40
Hz
Time
Amplitude (Intensity)


0:00 / 0:04


L1 Strain 20 Onwards Elongated
Amplitude
Time (sec)


L1 Strain Cutoff 20
Hz
Time
Amplitude (Intensity)


H1 L1 Superimposed
H1
L1
Strain * 1.e21
1.0
0.5
0.0
−0.5
−1.0
0.25
0.30
0.35
0.40
0.45
Time (sec)


H1 L1 Filtered Superimposed
H1
L1
Strain * 1.e21
1.00
0.75
0.50
0.25
0.00
−0.25
−0.50
−0.75
−1.00
0.25
0.30
0.35
0.40
0.45
Time (sec)

Results: Part 2

Now, although there are a lot of graphs and plots above, these images are mainly depicting just a few things: the effect Fourier Filtering has on this data, and what gravitional waves sound like as audio.

Listening to the audio, it is clear that it sounds like a wobbling frequency, as expected. Specifically, its wobbling at a low frequency, as depicted by the spectrogram plots where the intensity of the audio is much greater towards the bottom of the spectrum than the top. Additionally, the less impotant frequencies removed by the Fourier Transforms are the higher frequencies towards the top of the spectrum, with a blue color corresponding to a lesser intensity.

As for filtering the actual signal, there seems to be very little purpose in doing so here, simply because the data isn't jumping all over the place. However, there could be some merit to using Fourier filtering here to smooth the data out for a person looking at a large amount of this type of data and who is more interested in discovering the trends or patterns of the curves.

Pictured right above are the H1 and L1 strain signals superimposed atop one another. Here, they are quite difficult to really compare to each other, though the trend is apparent. A filtered version of them as shown right underneath provides an easier plot to look at for really comparing the two signals and noting the similarities and differences.

Of course, both signals are not the same because there was a time delay between one LIGO detecting gravitaional waves and the other detecting them, among other factors.

Methodology: Part 3

The entire project culminates to this part, where the Fourier Transform is used to classify audio. Several different attempts were made to create a classifier, but only the final is presented here, with a brief overview of the others.

Initially, a machine learning approach using scikit was attempted. Mainly due to a lack of experience with machine learning libraries and techniques, this attempt made it only as far as to classify different pitches in audio using kmeans clustering. Though interesting in and of itself, a method to compare datasets to some learned data failed to come about.

Then, the focus shifted over to try and force a learned dataset and compare the difference in values with a new dataset. In this scenario, the A4, F3, and F4 signals were merged together into one audio file and their respective pitches were classified. This didn't work for two reasons:

1. What was perceived to be usable data simply wasn't. Combining the signals together cannot yield a proper, baseline dataset because the majority pitch may change. Or, it may not, and either way you have no way of getting clear data. Using this method is "noisy".
2. Again, likely due to a lack of working with this library, there was no apparent way to tell which pitch was classified as what from the kmeans clustering outputs

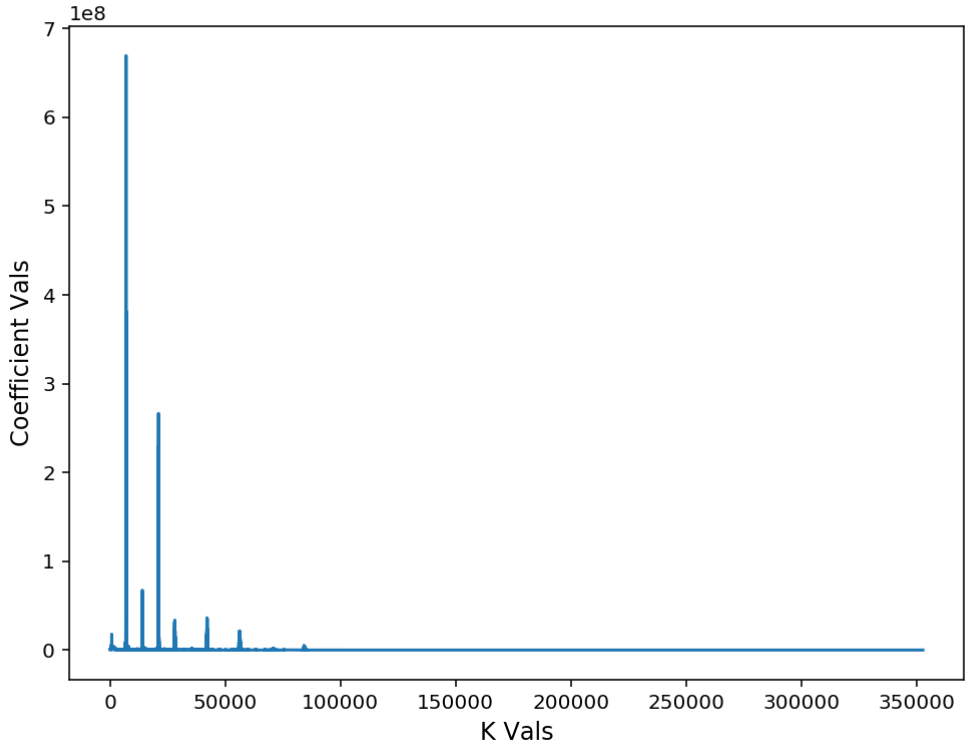
As such, the only valuable side-effect of attempting this method was gaining experience with scikit, and making pretty plots with plotly.

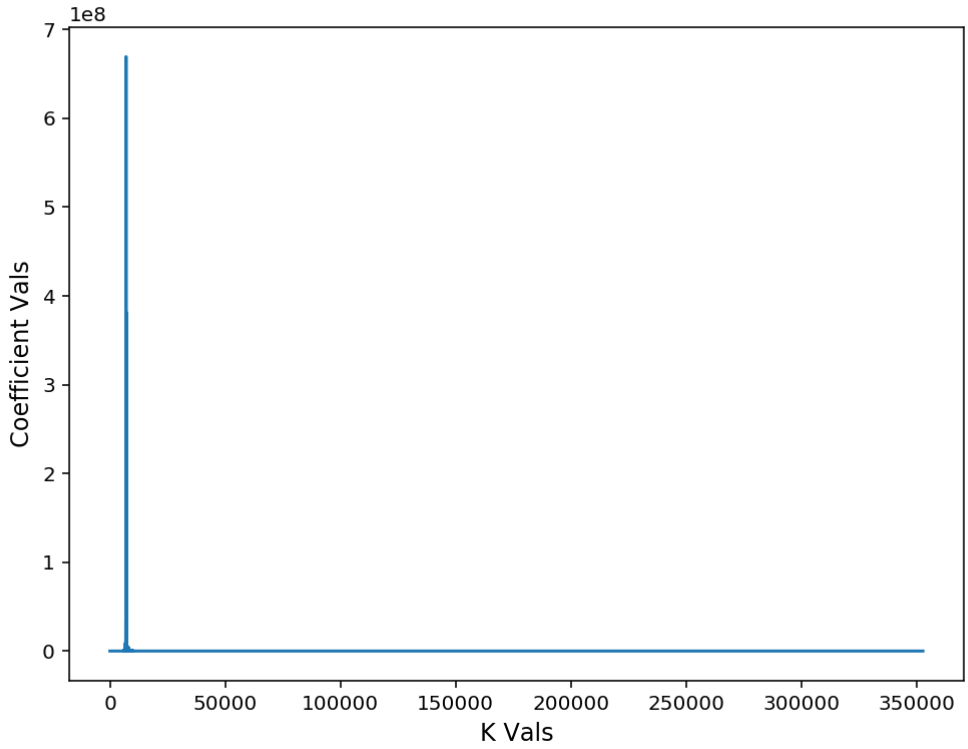
The previous method did have one good merit, however. That is, the idea of figuring out the difference between two datasets. Thus, instead of applying it through a machine learning approach, creating a "learned dataset" of sorts was done manually. For each of A4, F3, and F4 .wav files, their respective frequency domains were scrutinized in conjunction with their spectrogram plots in order to find the peaks corresponding to their baseline frequency. From a quick Google search, A4 sits at 440 Hz, F3 at 175 Hz, and F4 at 350 HZ. All peaks except the tallest were removed from the frequency domain and then the spectrogram was viewed again, yielding that the peak left was indeed the peak corresponding to the baseline frequencies.

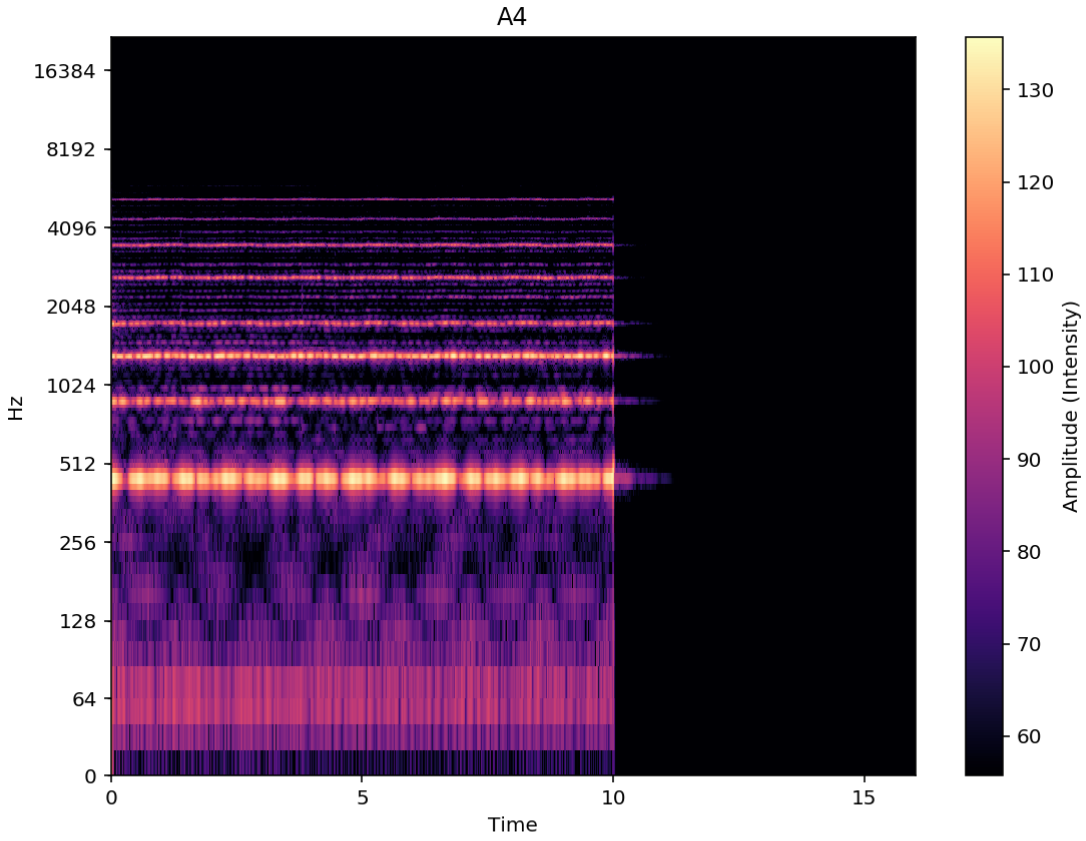
For some reason, however, this didn't apply to the .wav file corresponding to F4. This is likely due to some variance in that audio file that caused 2 other, larger peaks to form. Nonetheless, the peak corresponding to the baseline frequency for F4 was found.

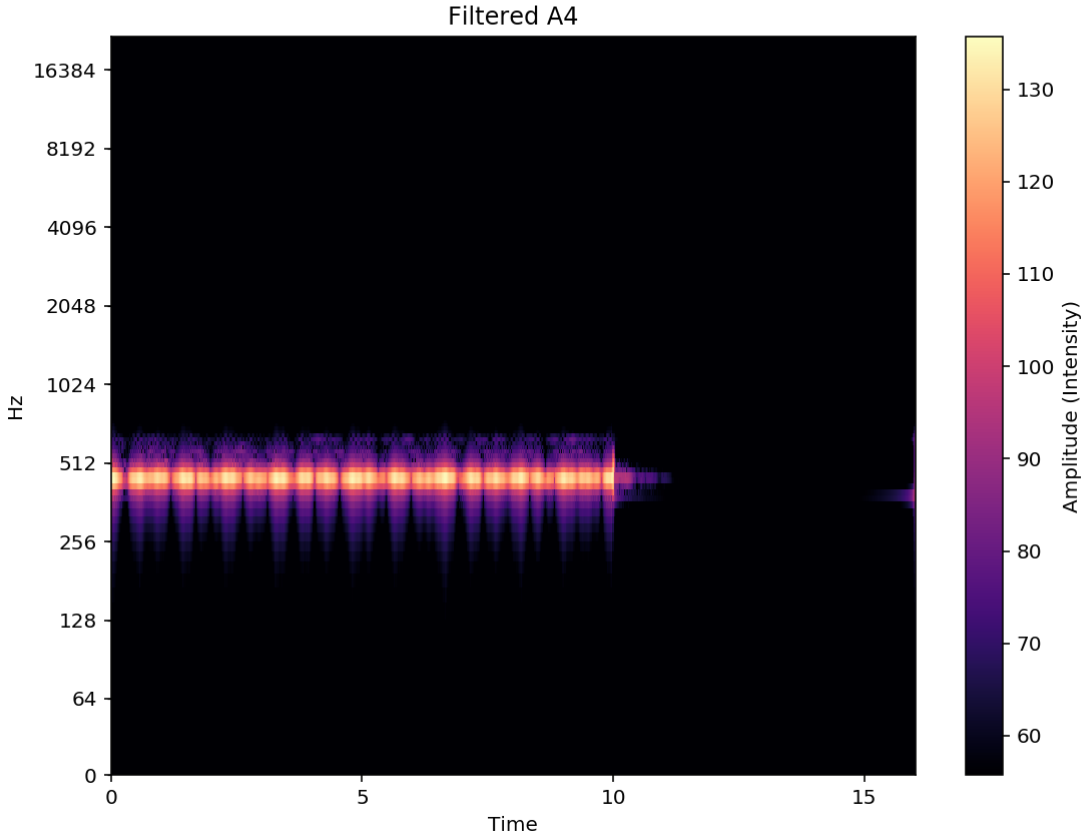
An average was taken for each of the peaks with respect to their location -- offsets of a few hundred on both sides. Now, in order to determine whether a given input .wav file is A4, F3, or F4, the file is matched to the musical note with the closest peak to its maximal peak.

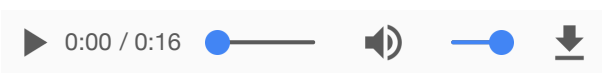
Below is the method in which the averages were computed; simply by hand after using %matplotlib notebook to look closer into the frequency domain.

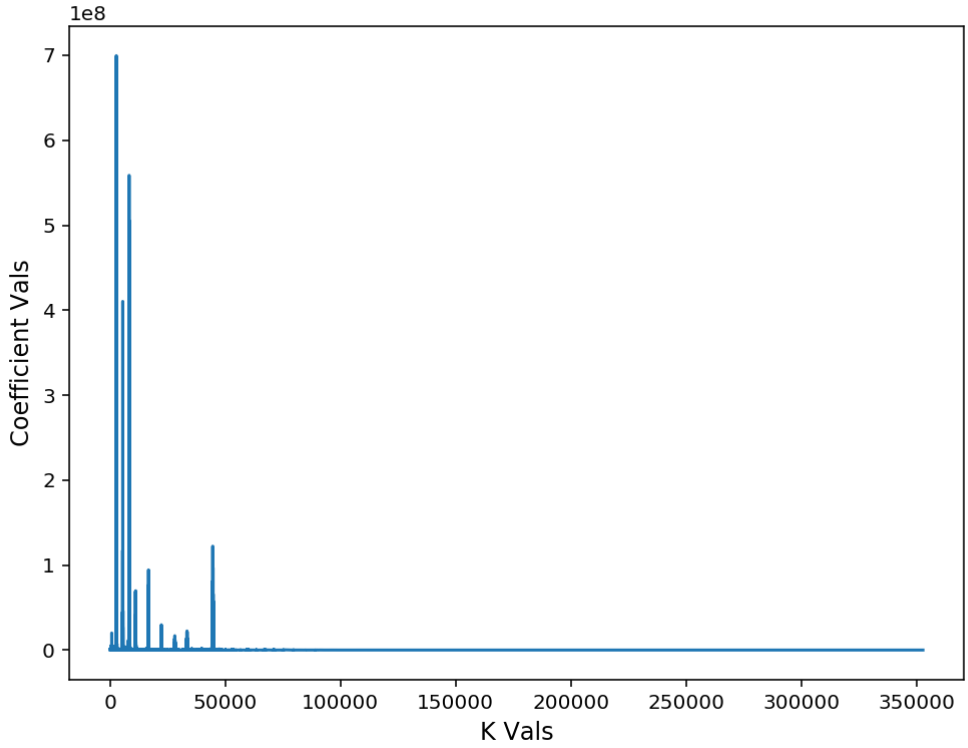

1e8
7
6
5
4
3
2
1
0
Coefficient Vals
0
50000
100000
150000
200000
250000
300000
350000
K Vals

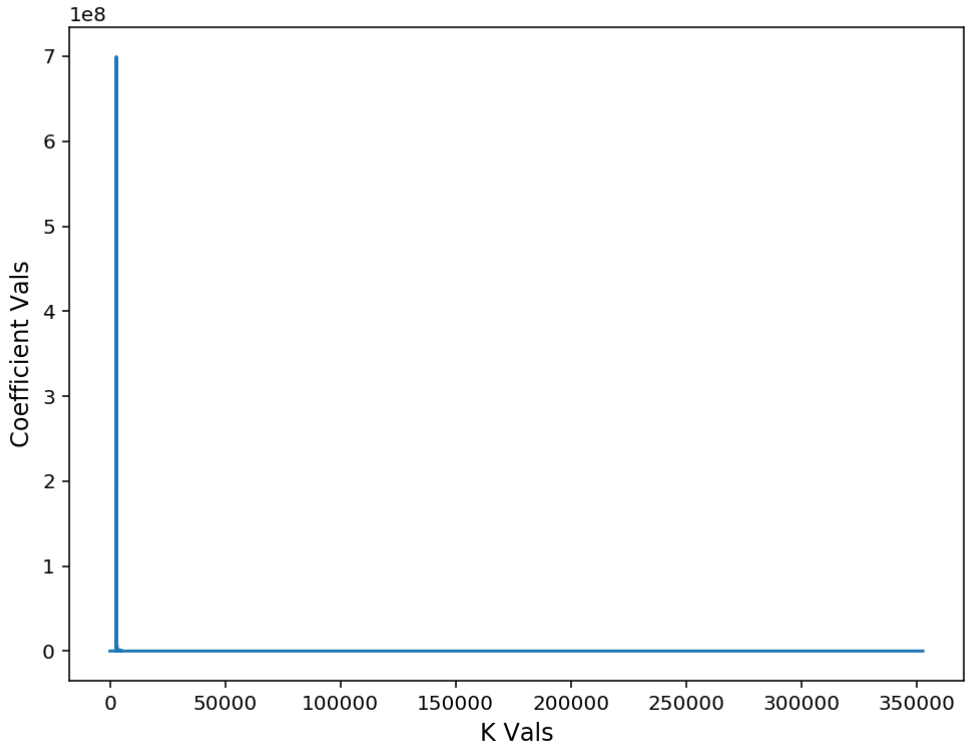

1e8
7
6
5
4
3
2
1
0
Coefficient Vals
0
50000
100000
150000
200000
250000
300000
350000
K Vals

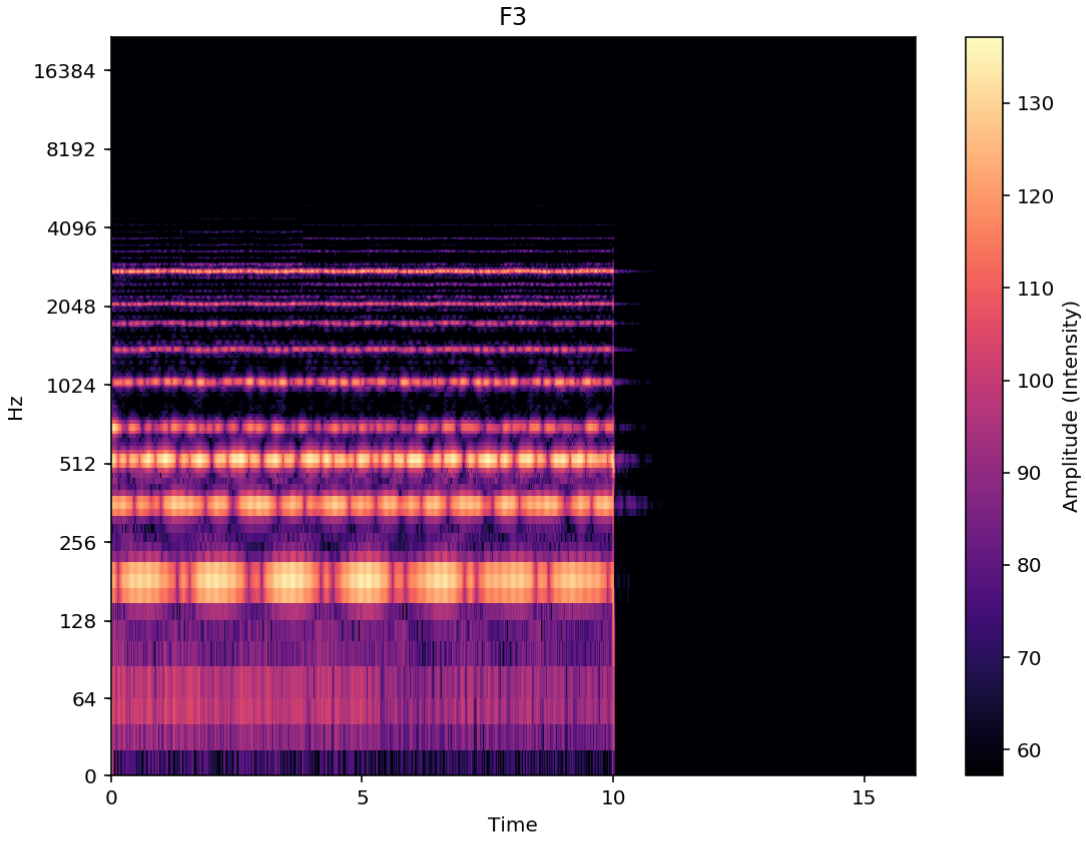

A4
16384
8192
4096
2048
1024
512
256
128
64
0
Hz
0
5
10
15
Time
130
120
110
100
90
80
70
60
Amplitude (Intensity)

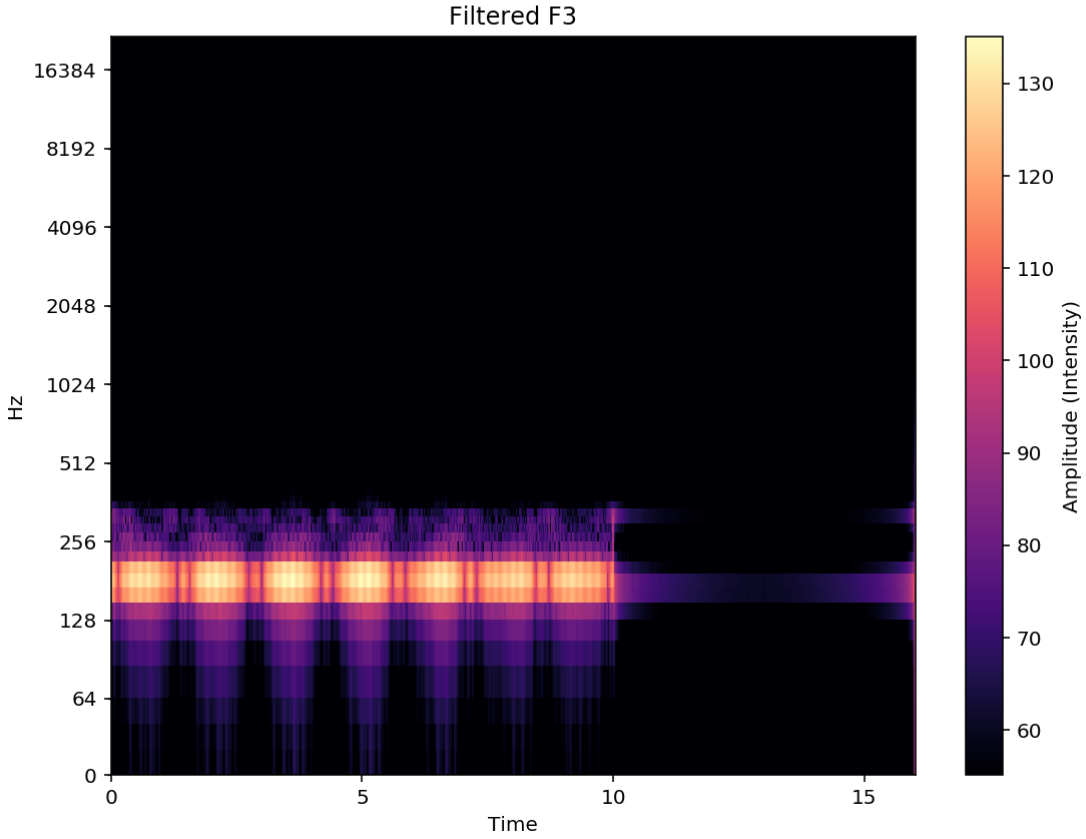

Filtered A4
16384
8192
4096
2048
1024
512
256
128
64
0
Hz
0
5
10
15
Time
130
120
110
100
90
80
70
60
Amplitude (Intensity)

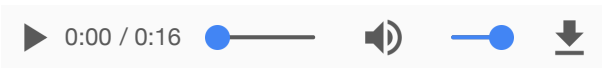

0:00 / 0:16

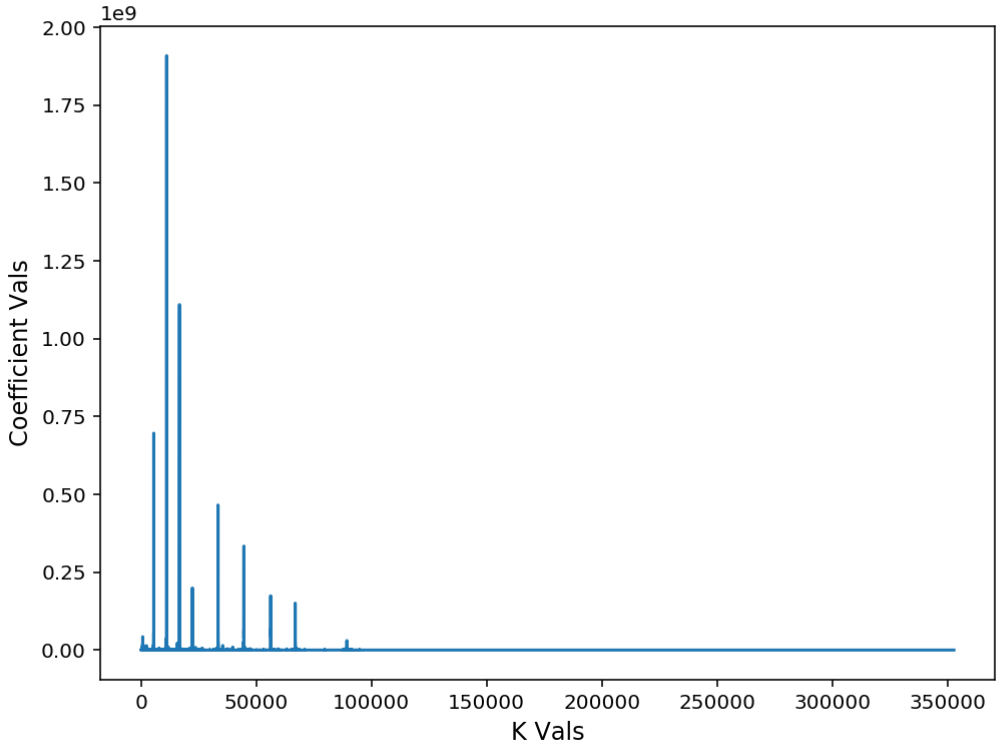

1e8
7
6
5
4
3
2
1
0
Coefficient Vals
0
50000
100000
150000
200000
250000
300000
350000
K Vals

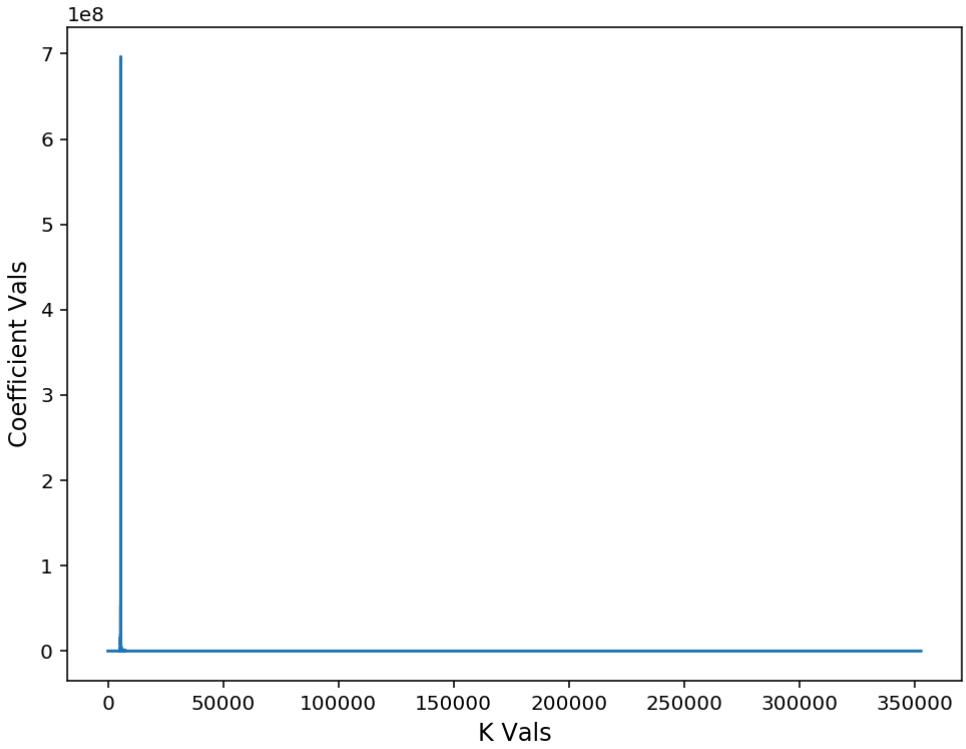

1e8
7
6
5
4
3
2
1
0
Coefficient Vals
0
50000
100000
150000
200000
250000
300000
350000
K Vals

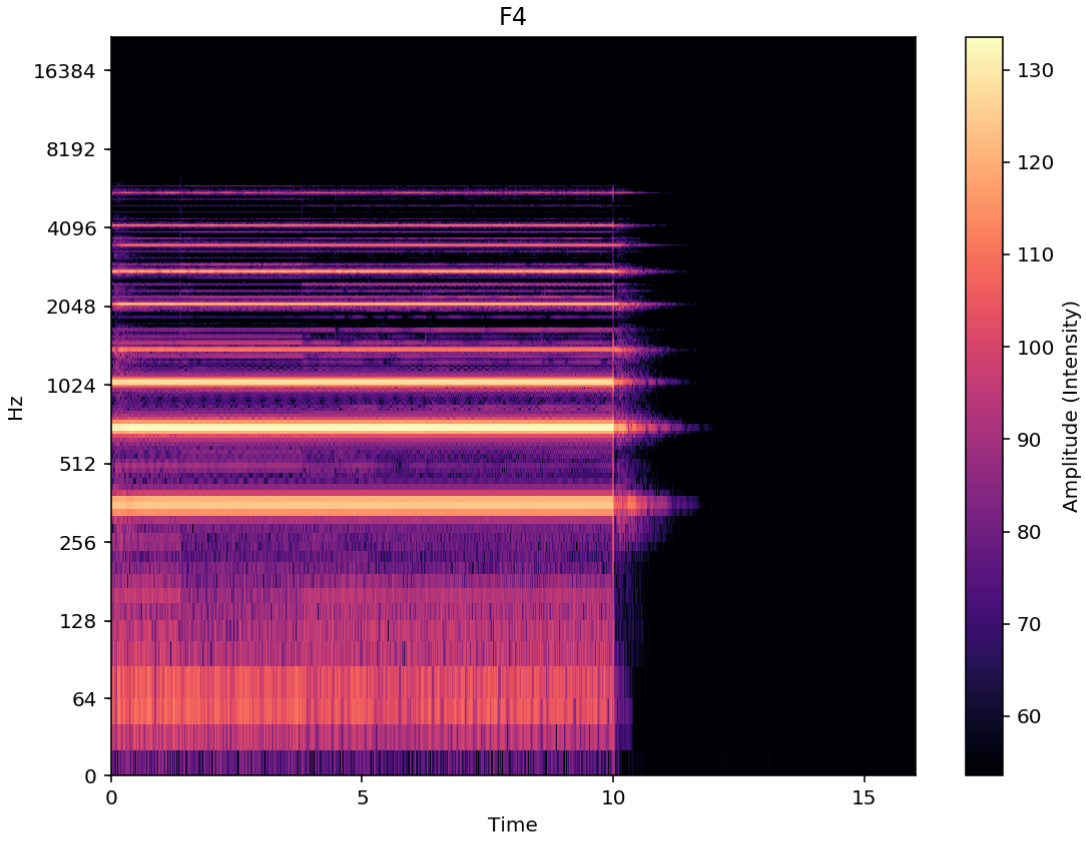

F3
16384
8192
4096
2048
1024
512
256
128
64
0
Hz
0
5
10
15
Time
130
120
110
100
90
80
70
60
Amplitude (Intensity)

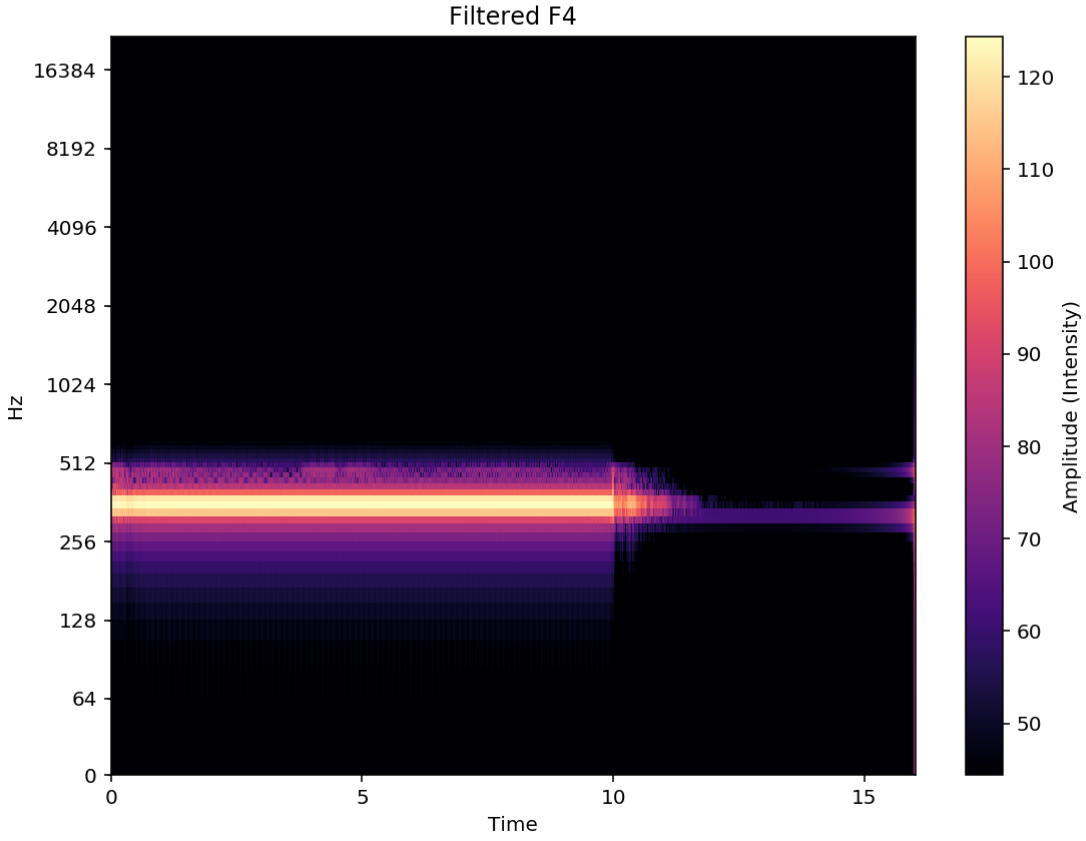

Filtered F3
16384
8192
4096
2048
1024
512
256
128
64
0
Hz
0
5
10
15
Time
130
120
110
100
90
80
70
60
Amplitude (Intensity)

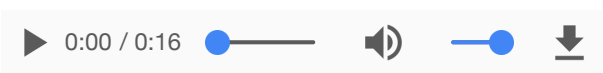

0:00 / 0:16


1e9
2.00
1.75
1.50
1.25
1.00
0.75
0.50
0.25
0.00
Coefficient Vals
0
50000
100000
150000
200000
250000
300000
350000
K Vals


1e8
7
6
5
4
3
2
1
0
Coefficient Vals
0
50000
100000
150000
200000
250000
300000
350000
K Vals

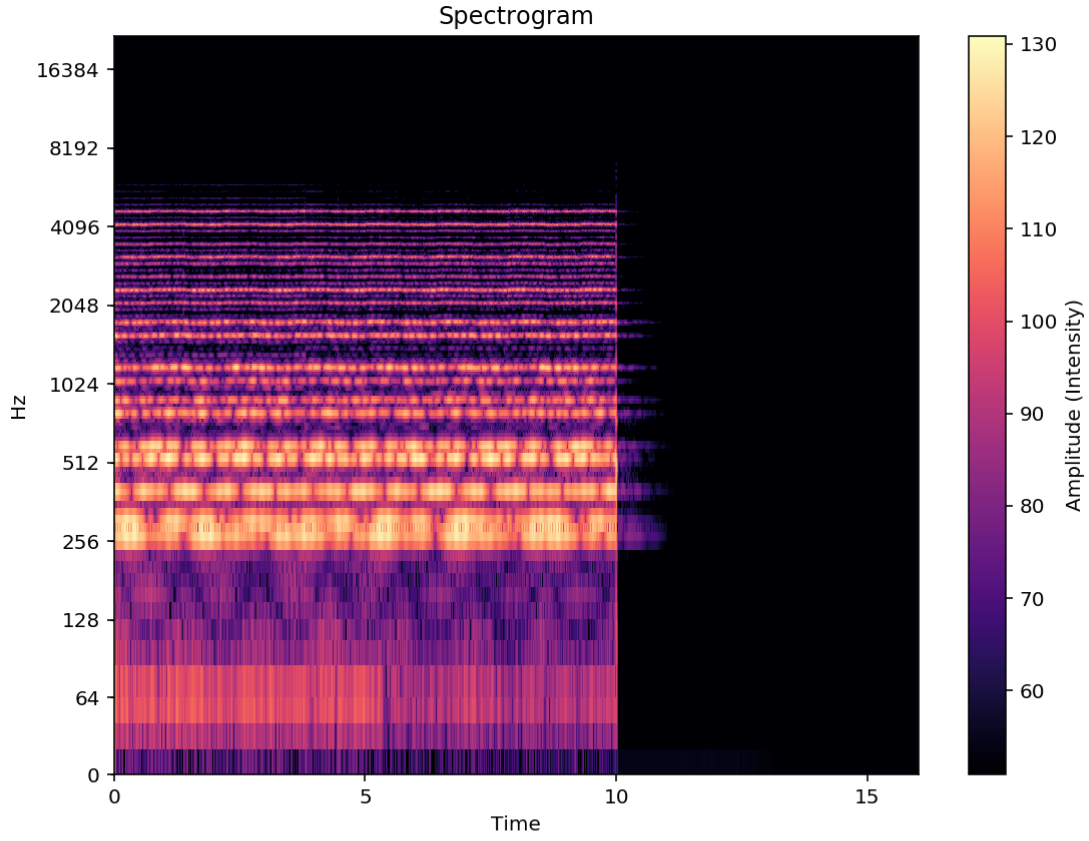

F4
16384
8192
4096
2048
1024
512
256
128
64
0
Hz
0
5
10
15
Time
130
120
110
100
90
80
70
60
Amplitude (Intensity)

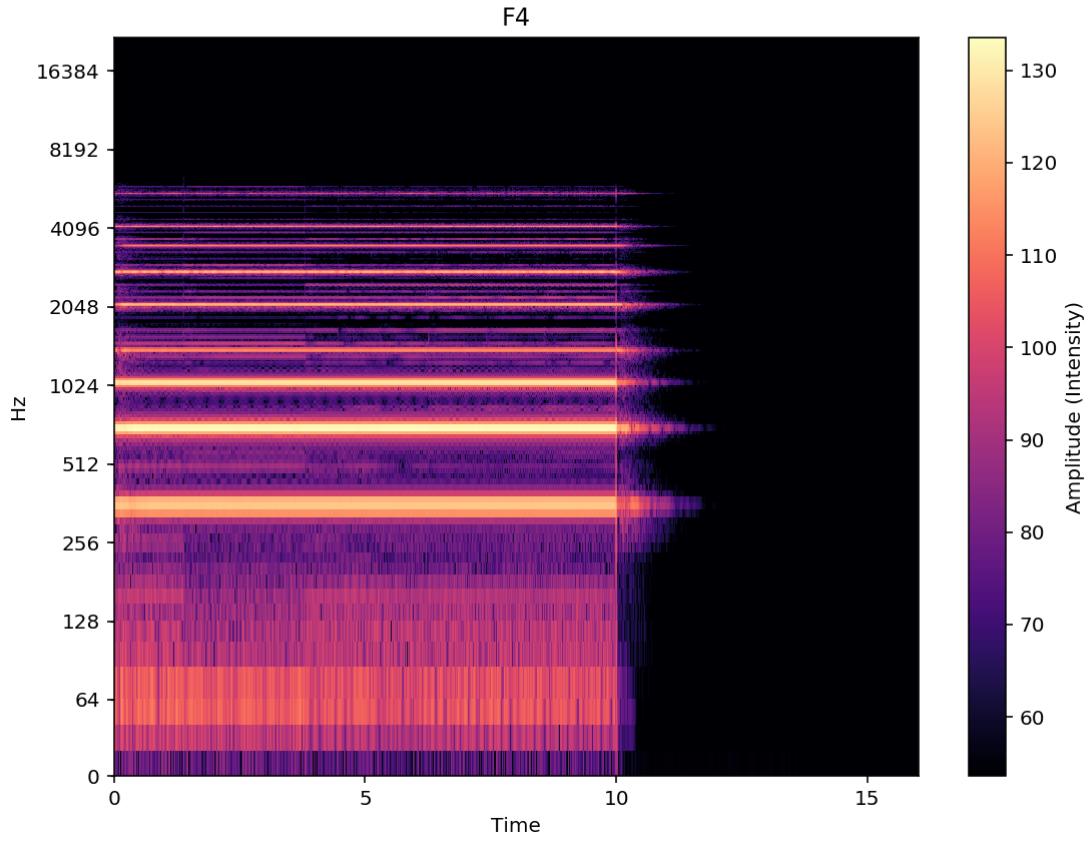

Filtered F4
16384
8192
4096
2048
1024
512
256
128
64
0
Hz
0
5
10
15
Time
120
110
100
90
80
70
60
50
Amplitude (Intensity)


0:00 / 0:16

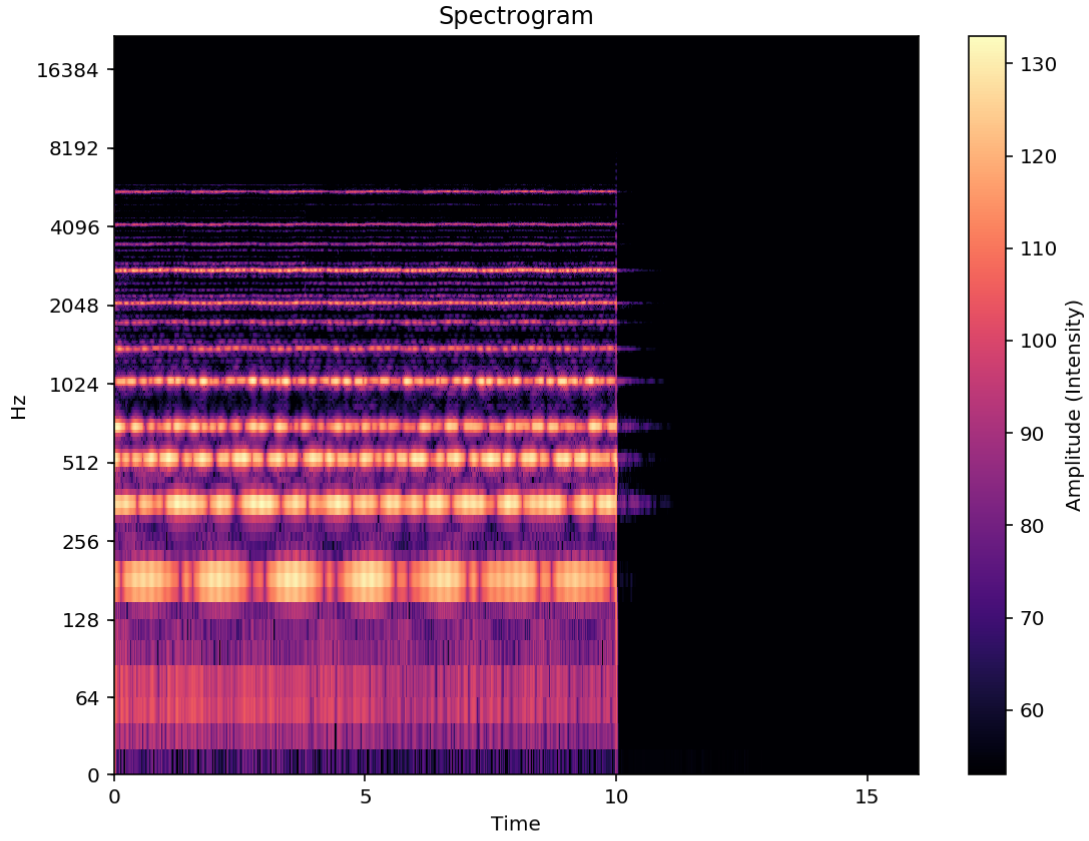
CLASSIFICATION RUNNING BELOW

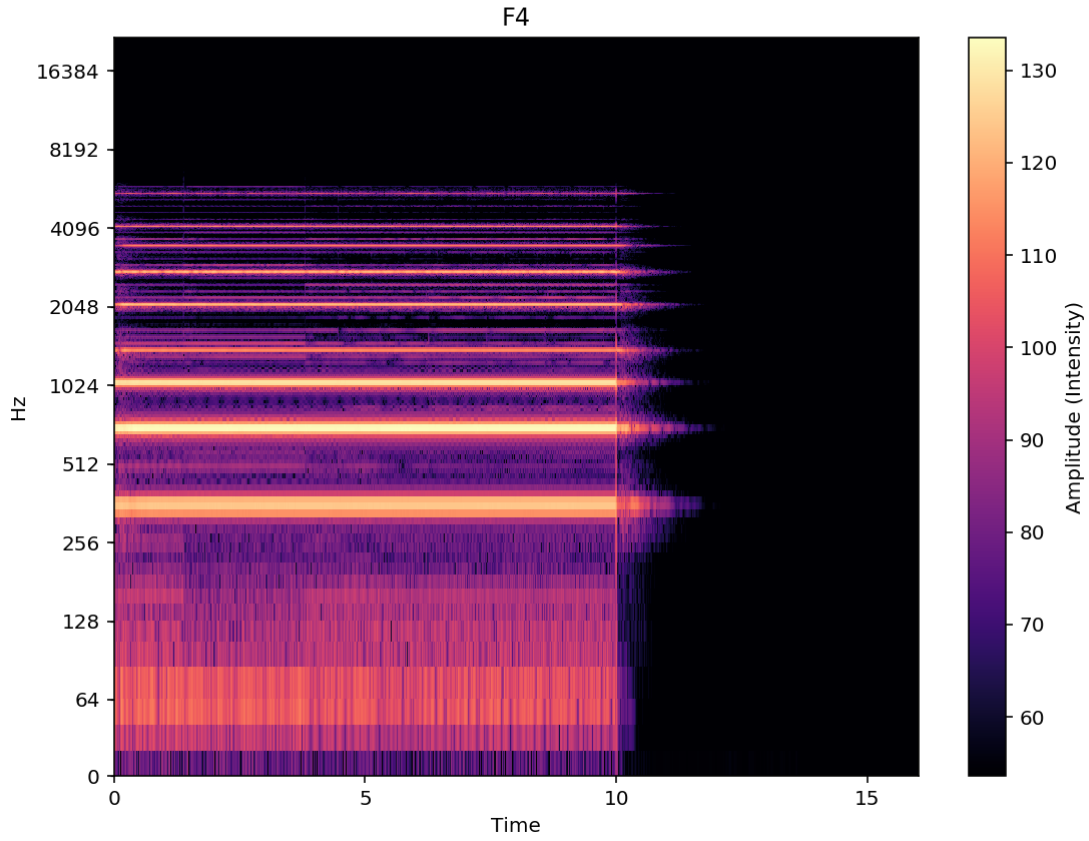
```
This audio file is classified as an F4 note
```



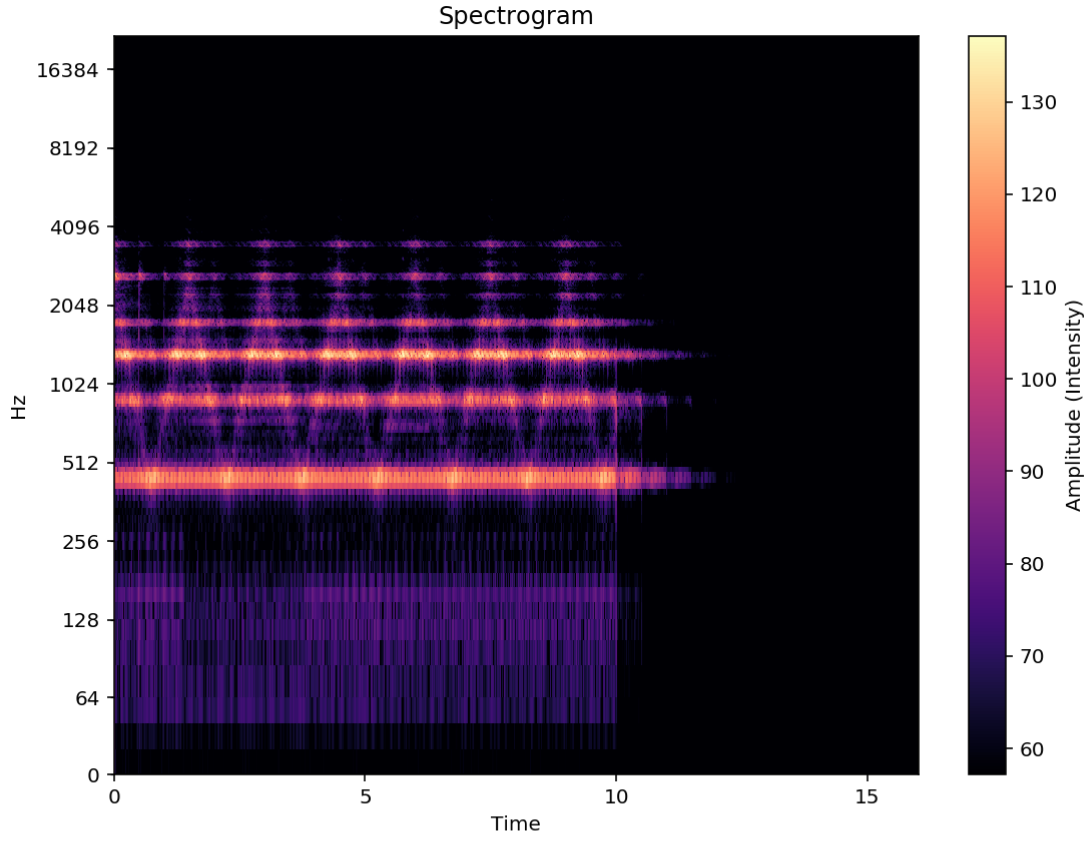


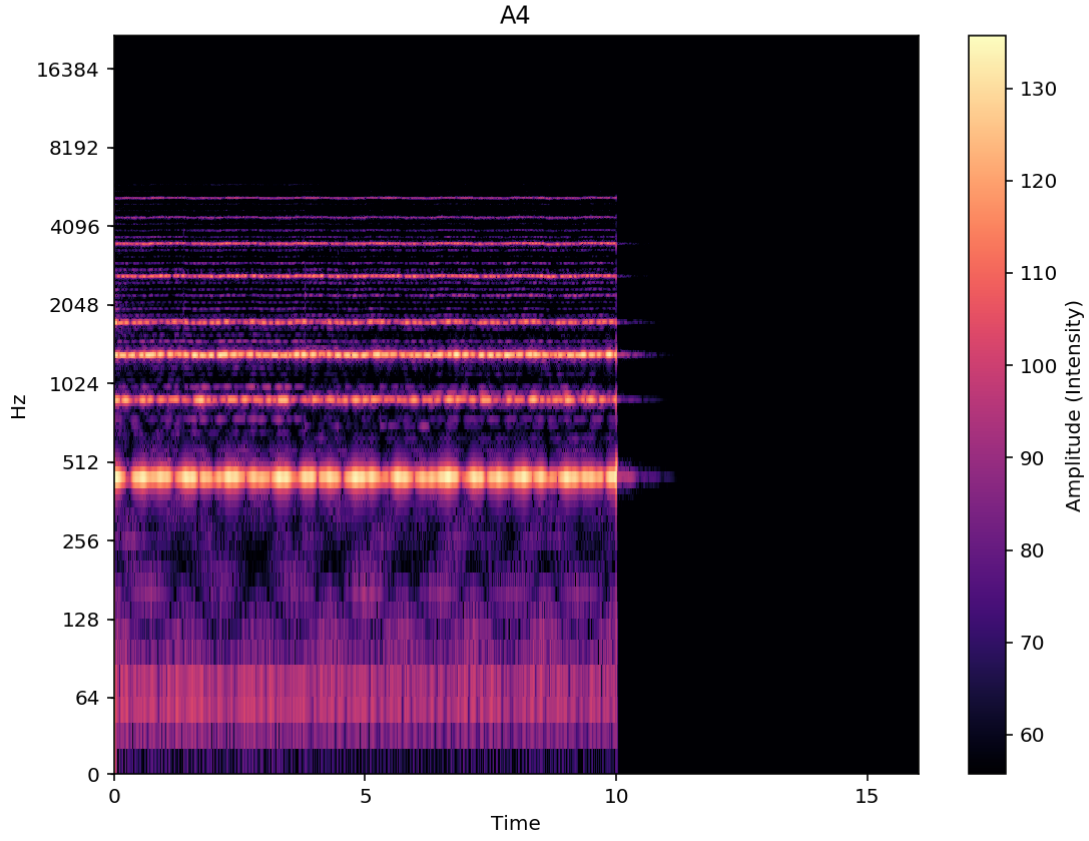
```
This audio file is classified as an F4 note
```



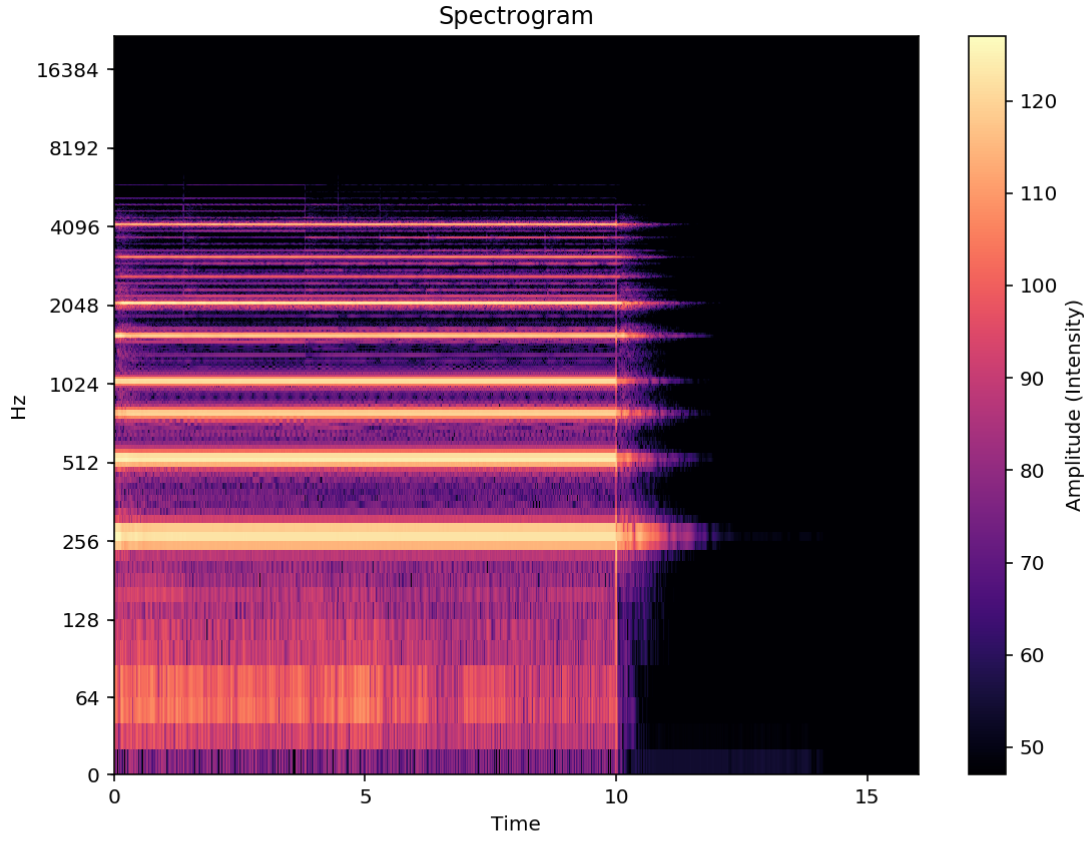


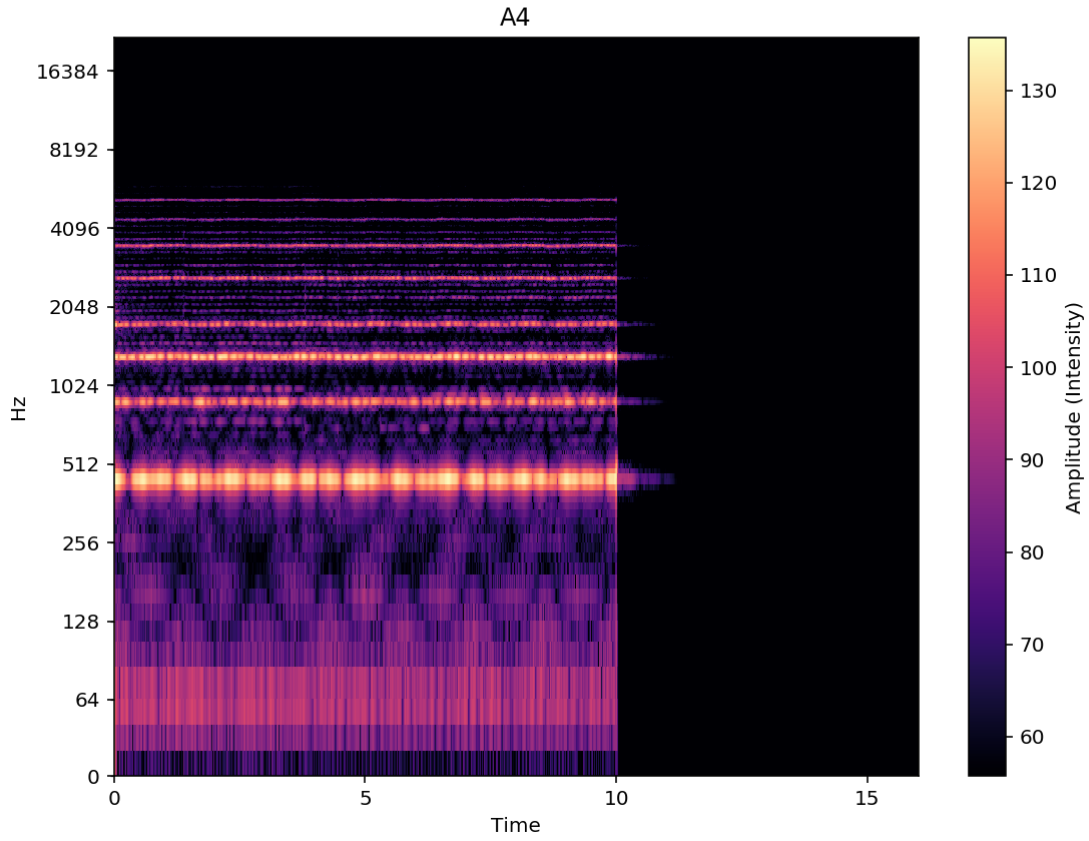
```
This audio file is classified as an A4 note
```



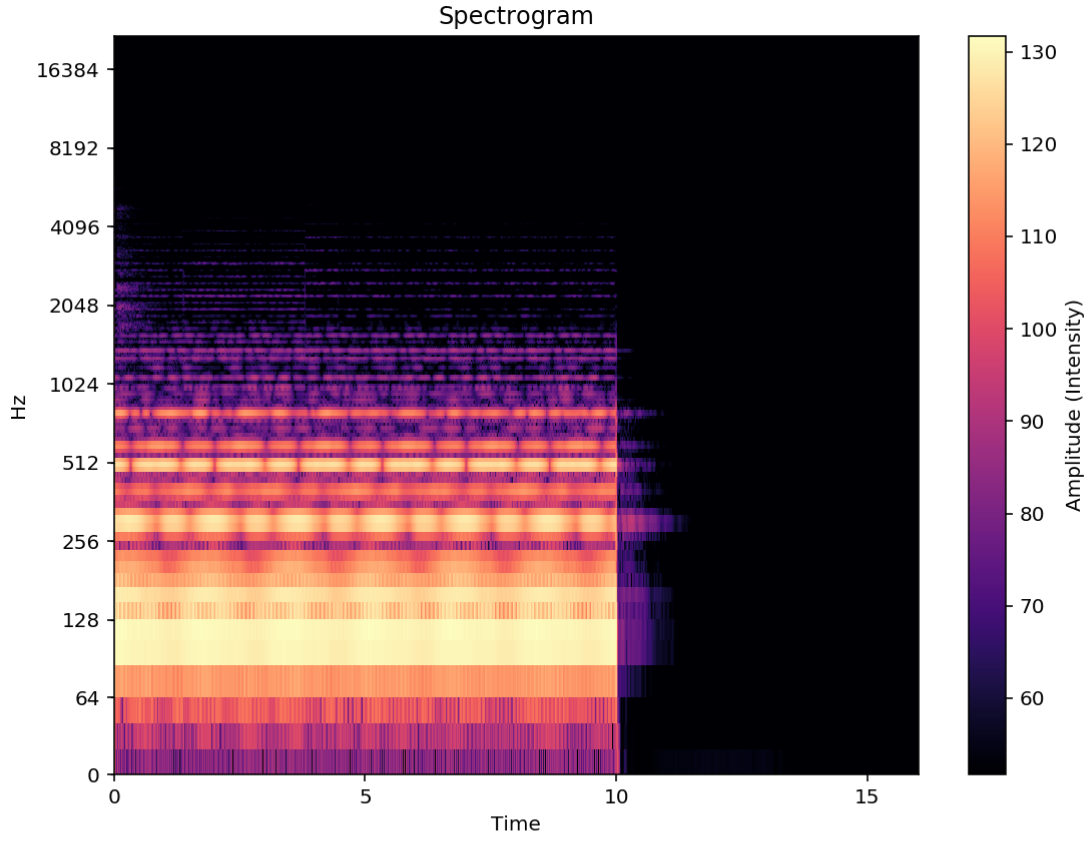


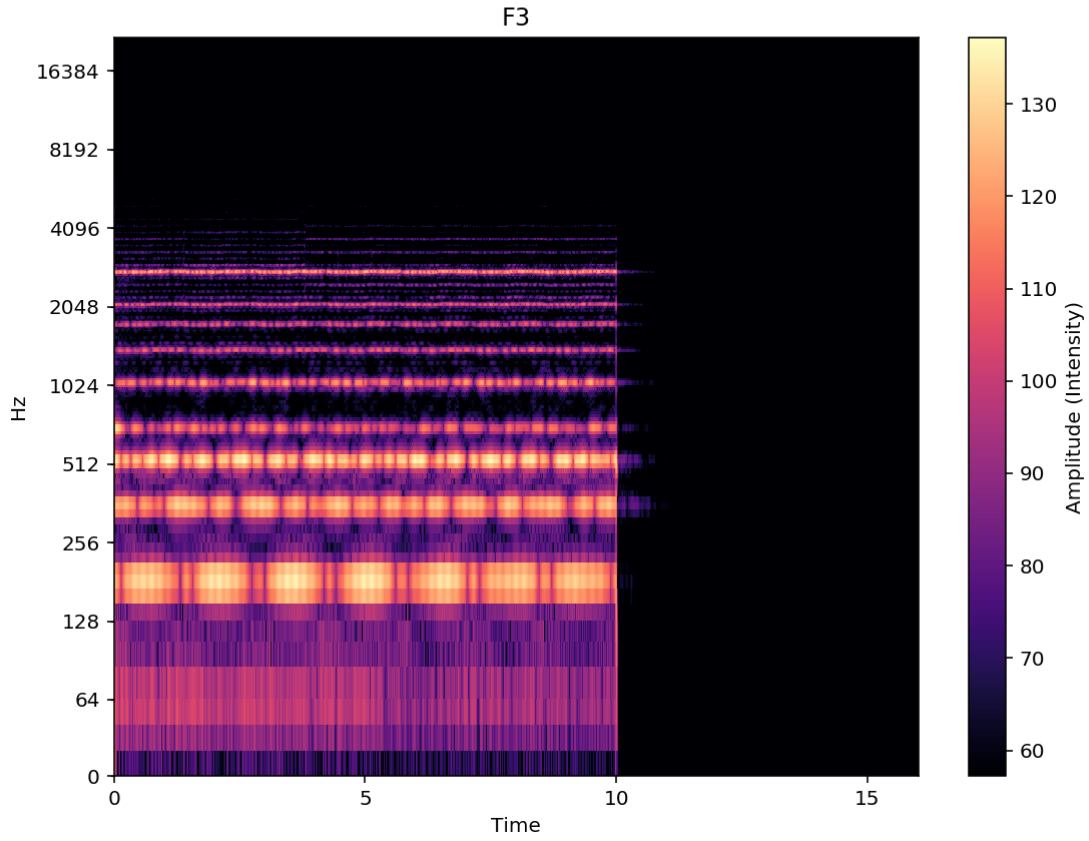
```
This audio file is classified as an A4 note
```



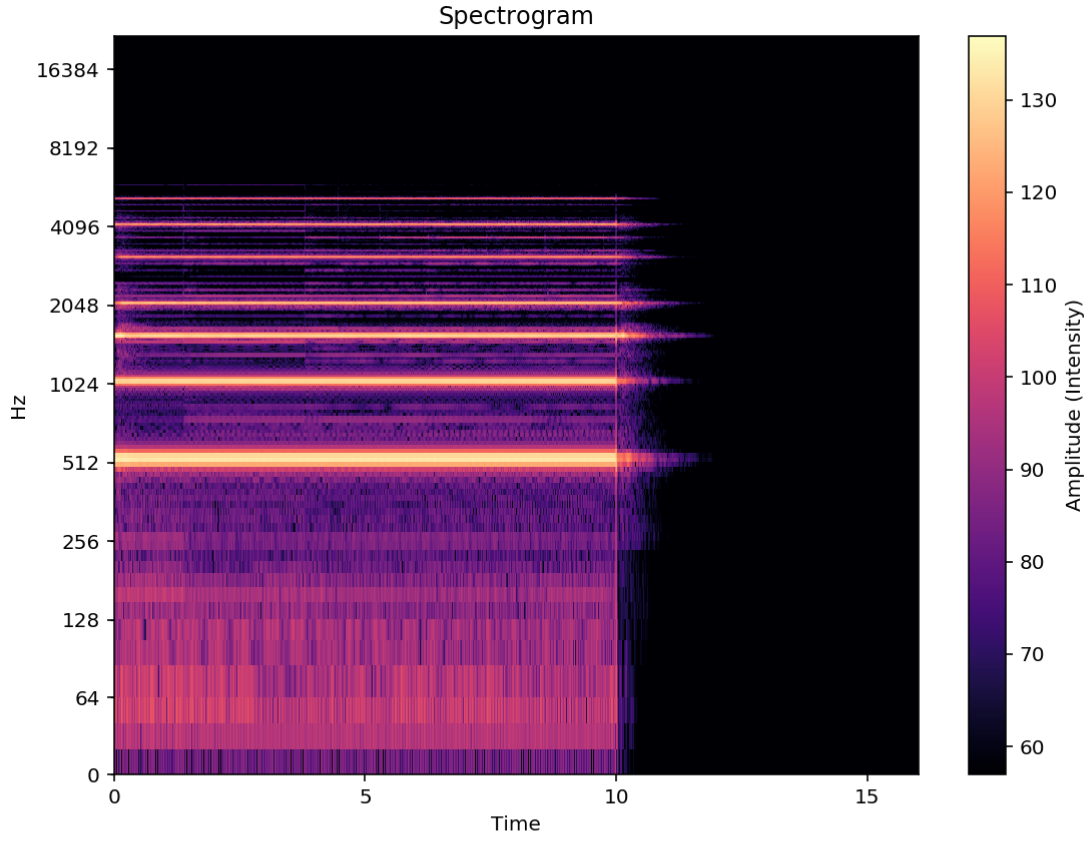


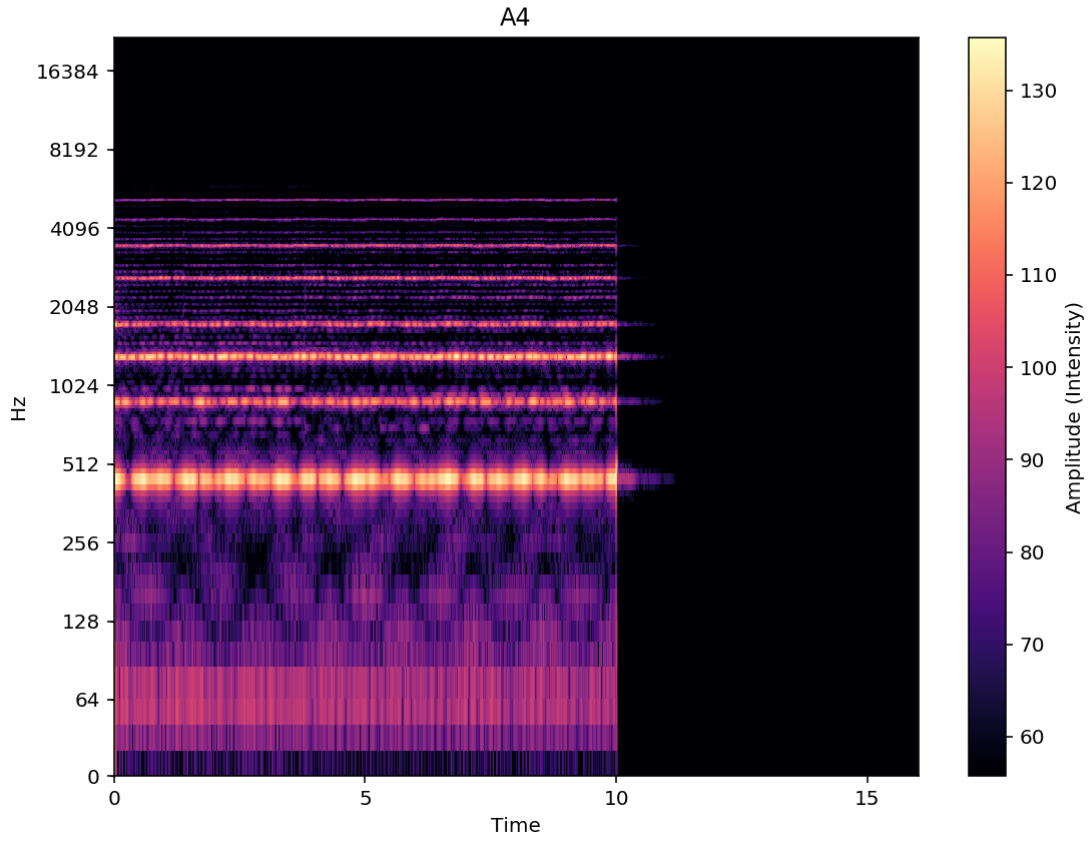
```
This audio file is classified as an F3 note
```



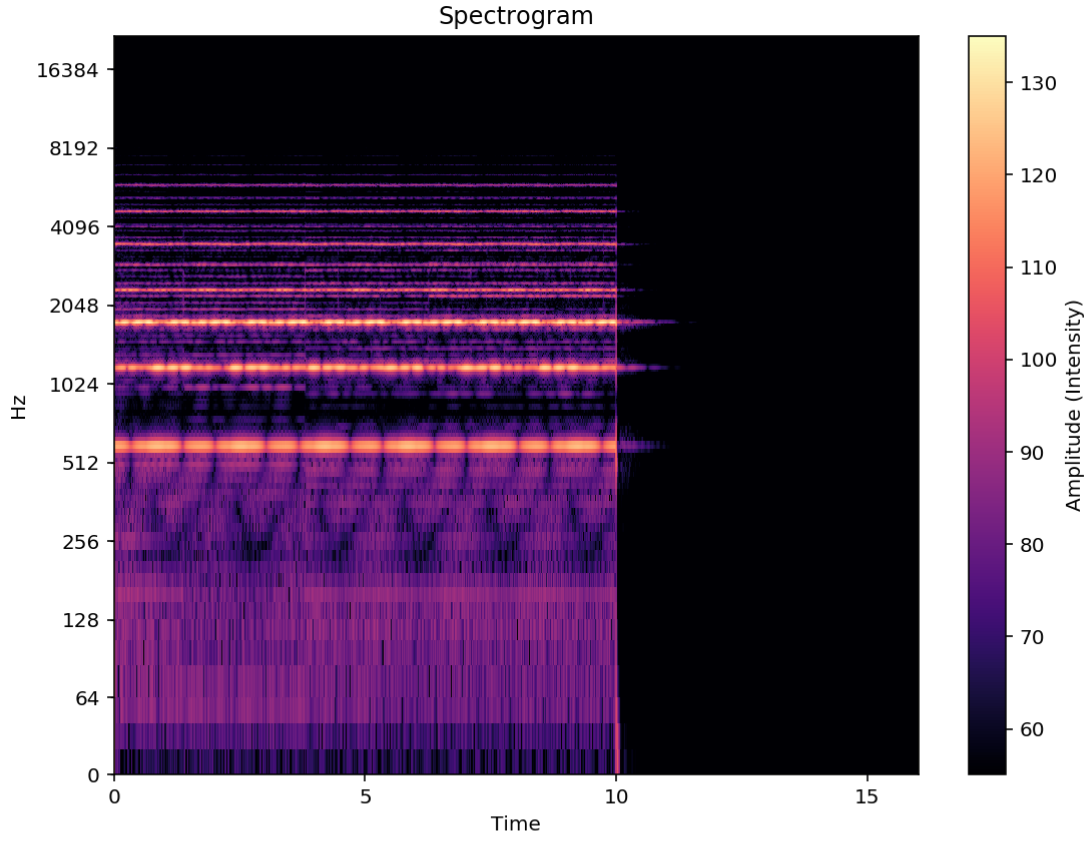


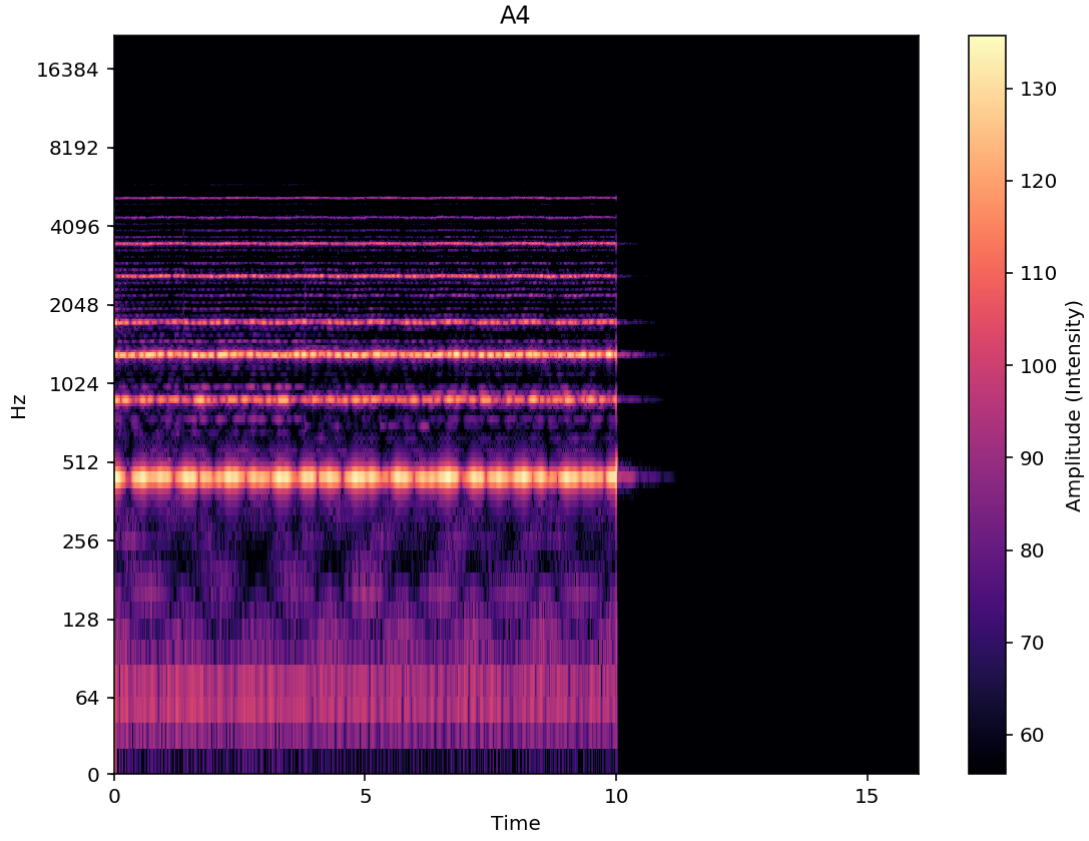
```
This audio file is classified as an A4 note
```



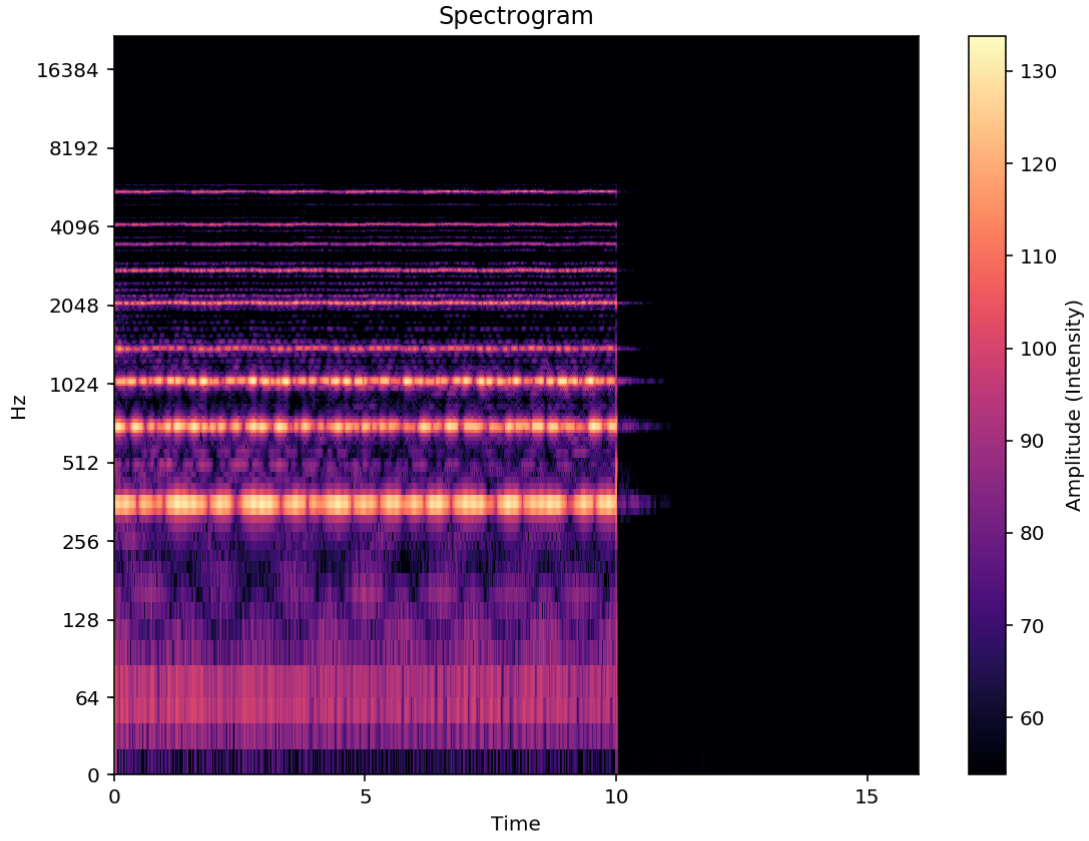


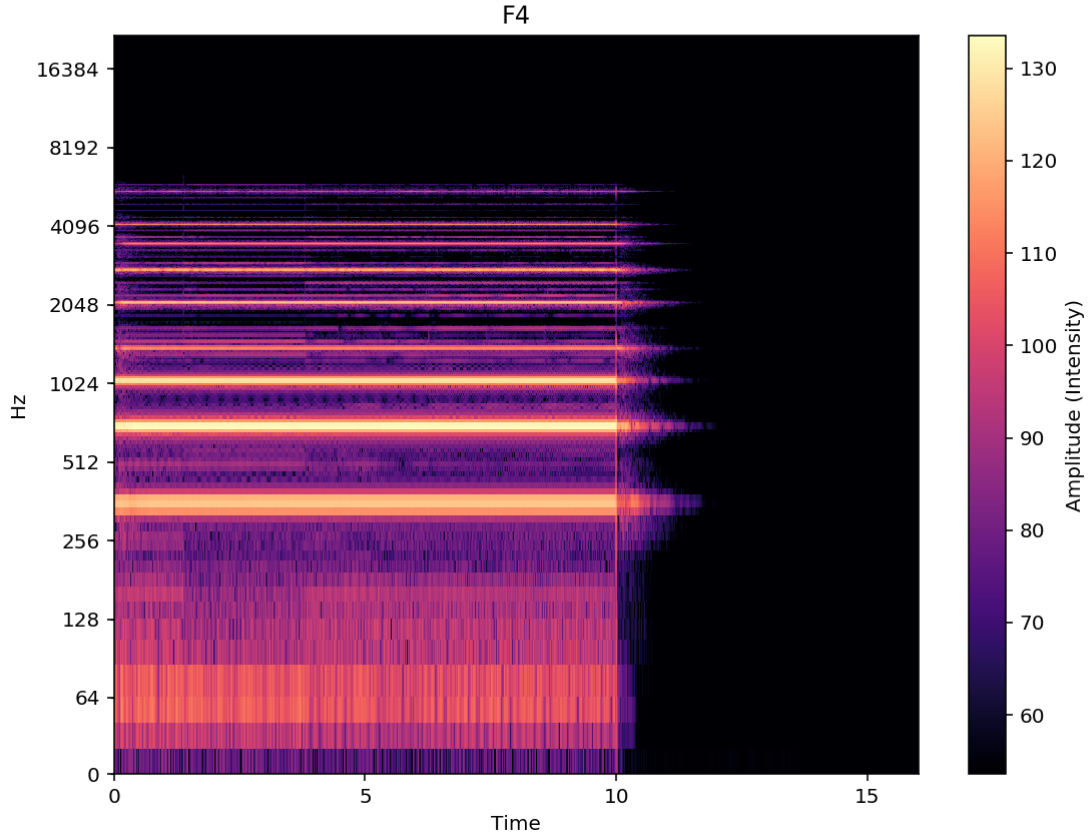
```
This audio file is classified as an A4 note
```



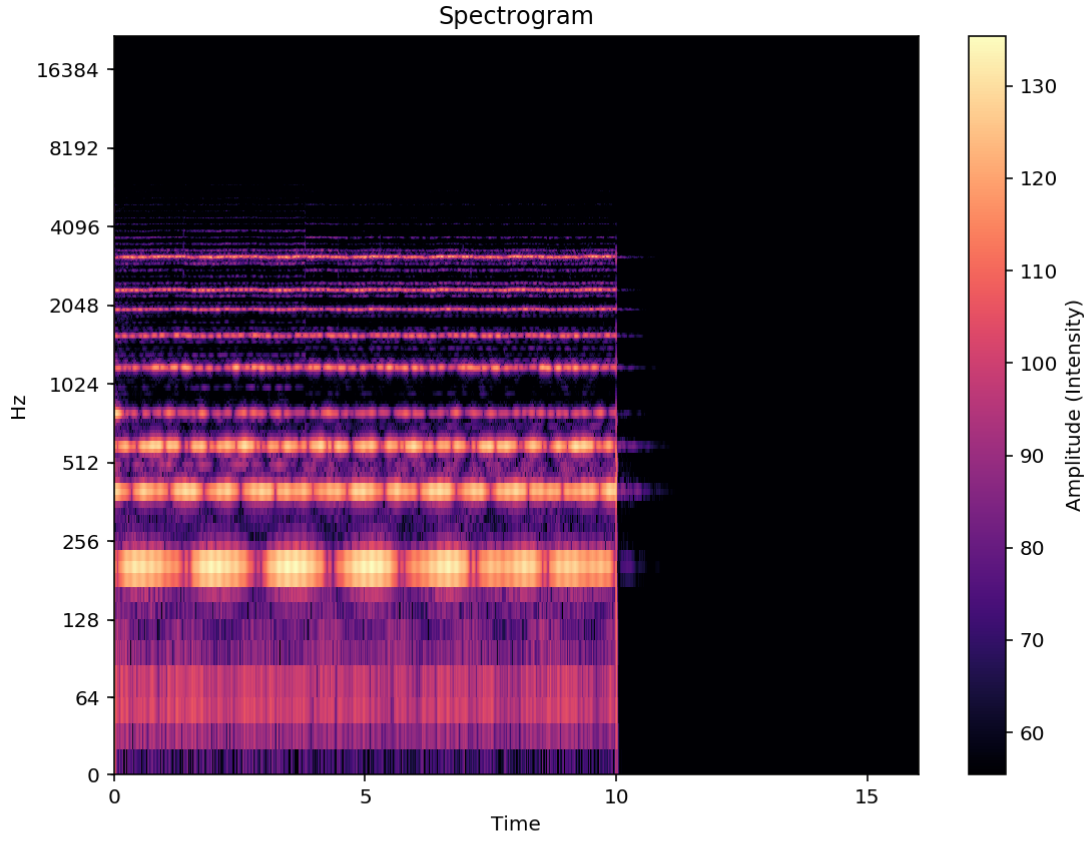


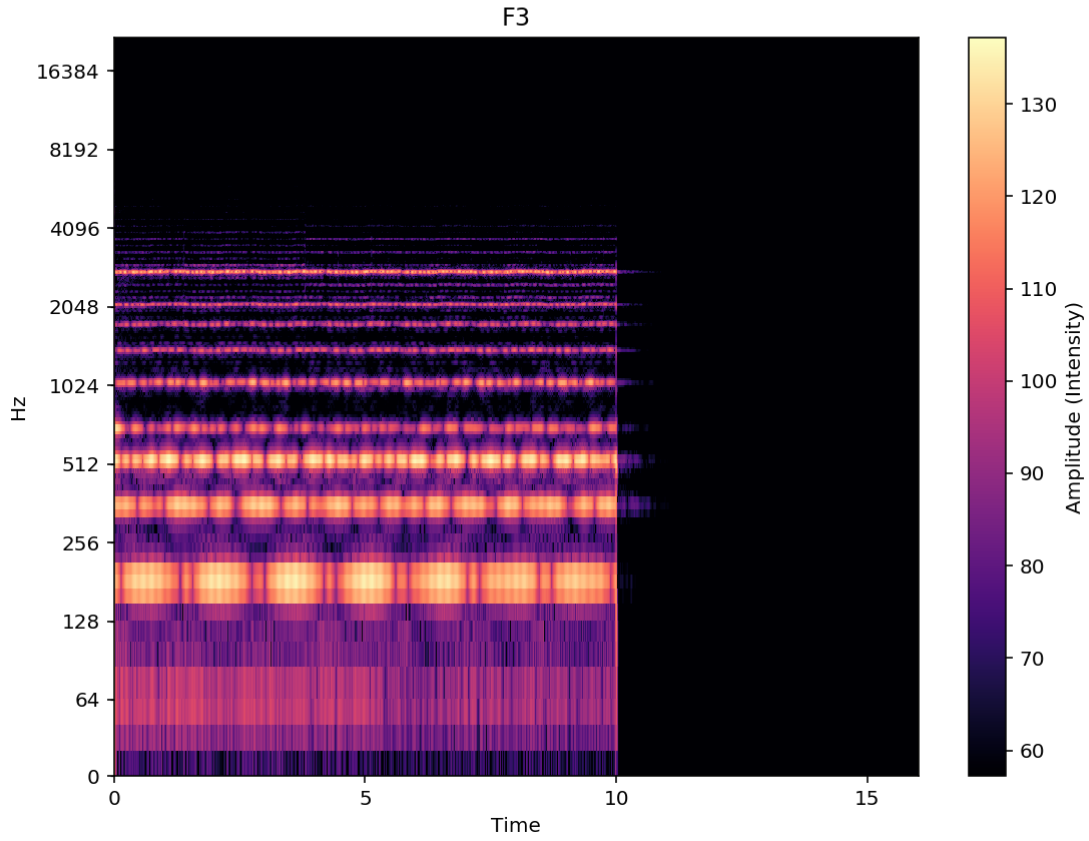
```
This audio file is classified as an F4 note
```



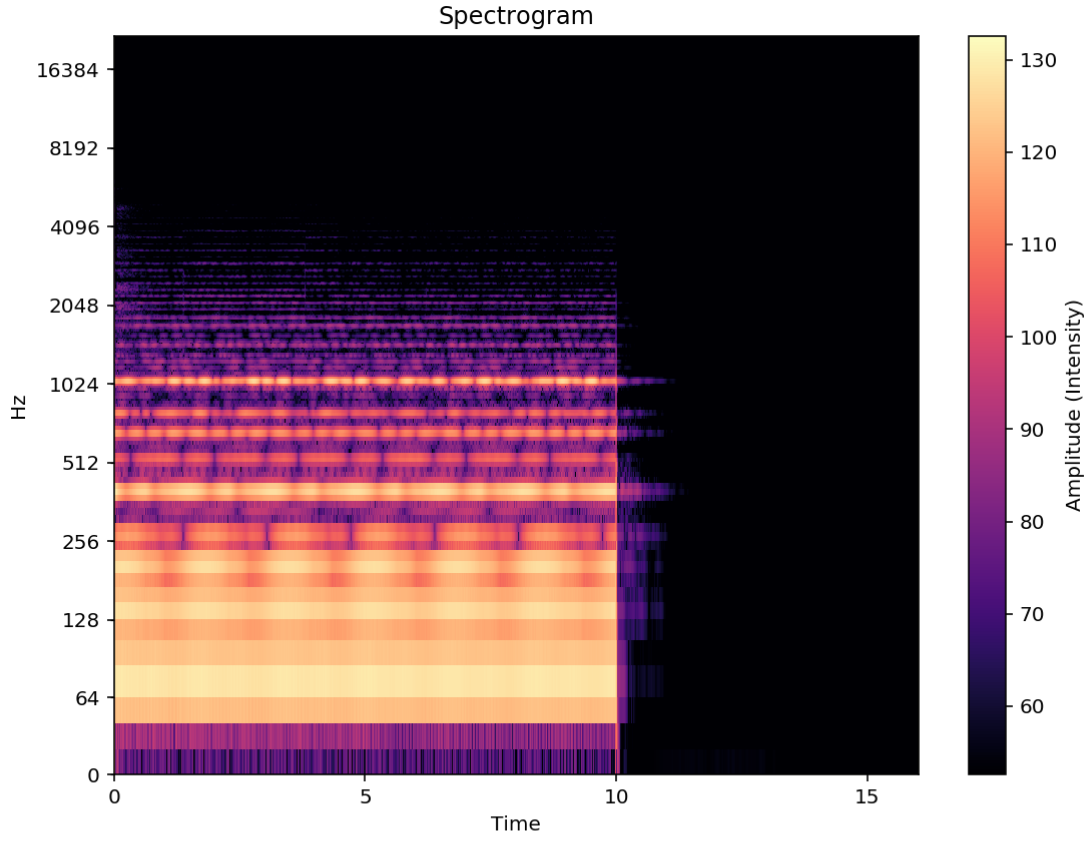


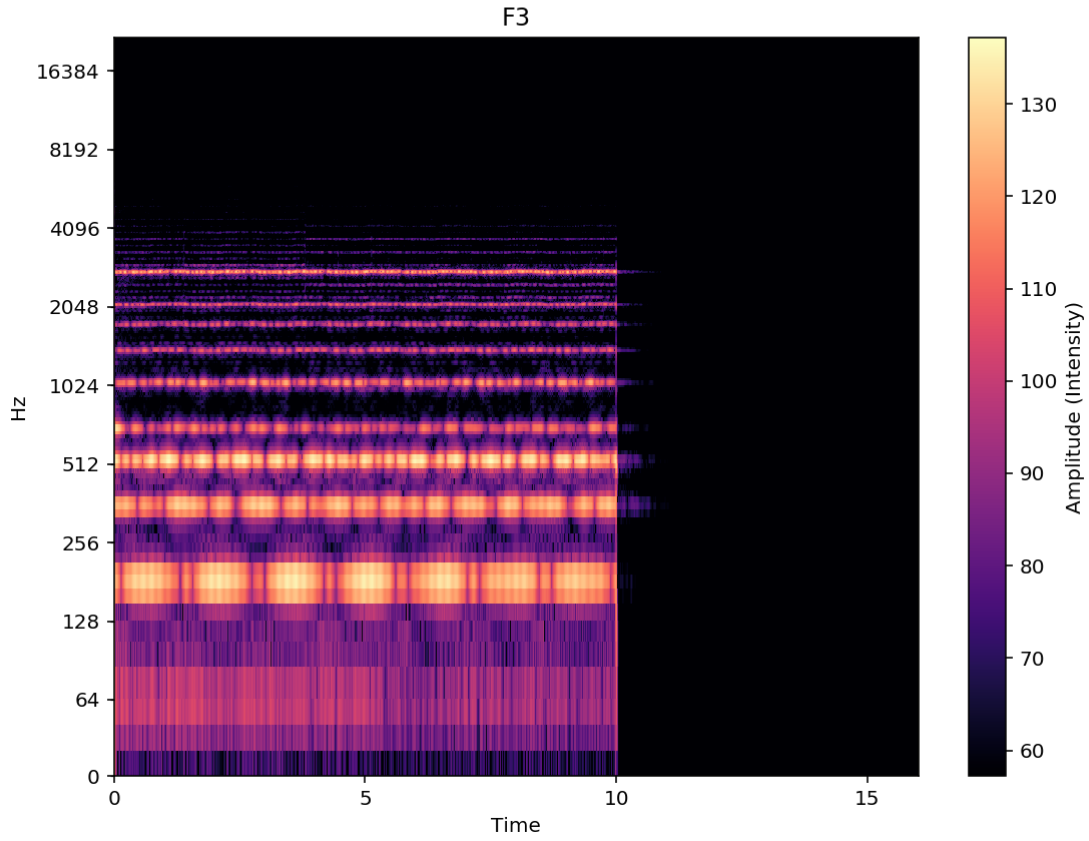
```
This audio file is classified as an F3 note
```



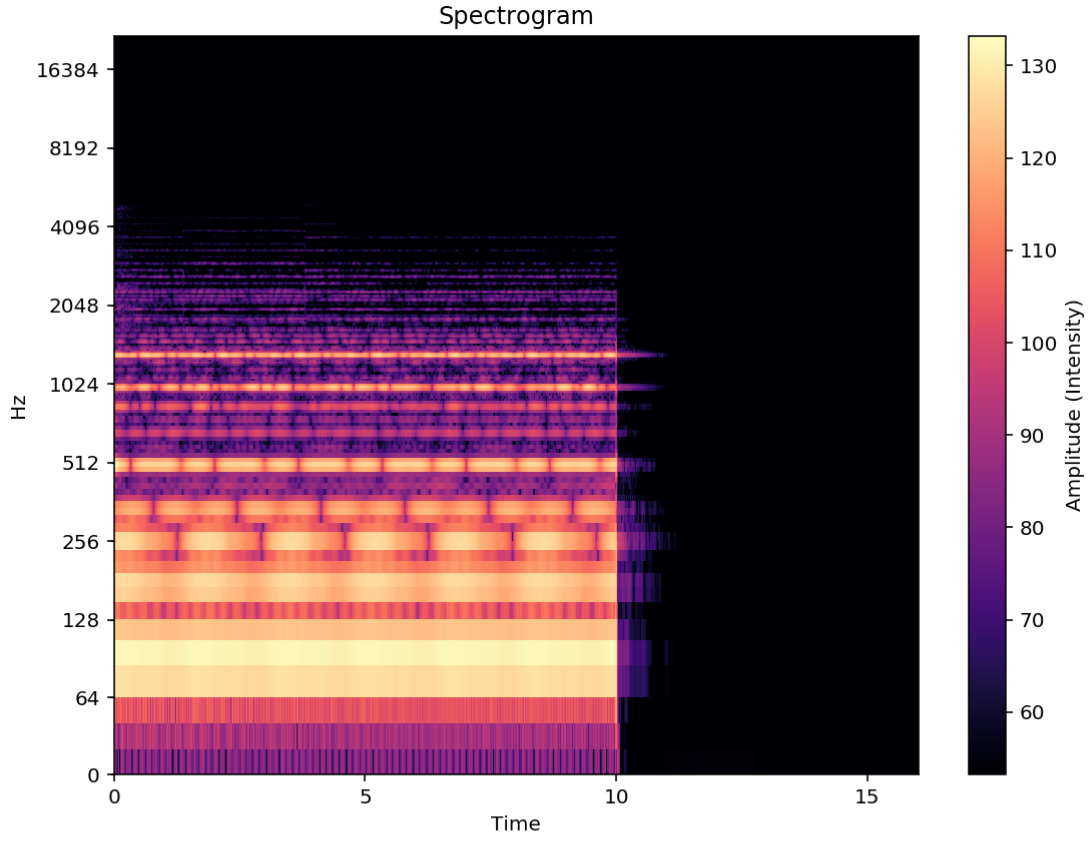


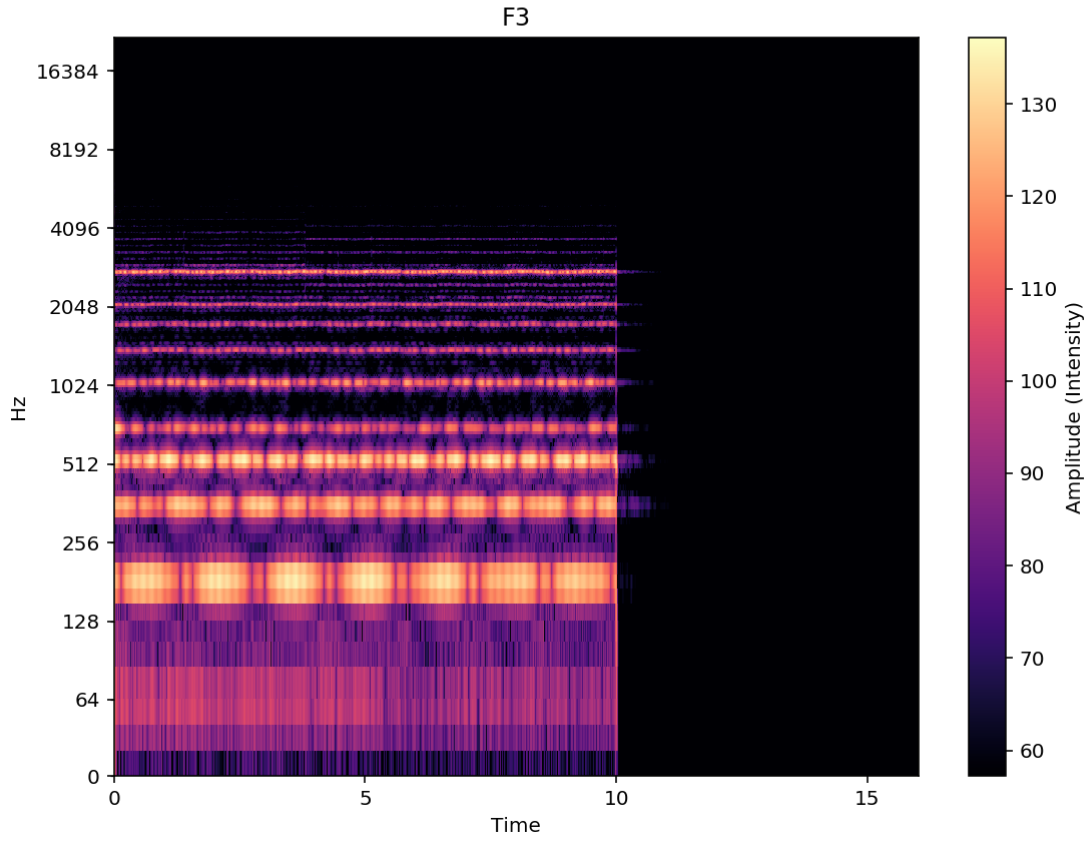
```
This audio file is classified as an F3 note
```





```
This audio file is classified as an F3 note
```





```
This audio file is classified as an F3 note
```