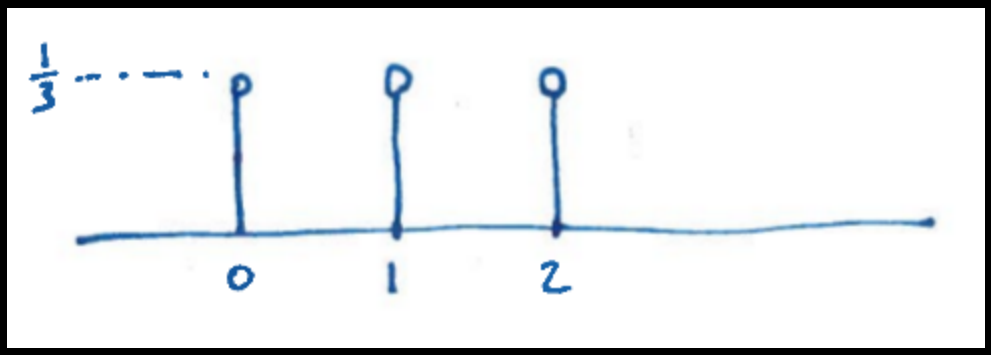
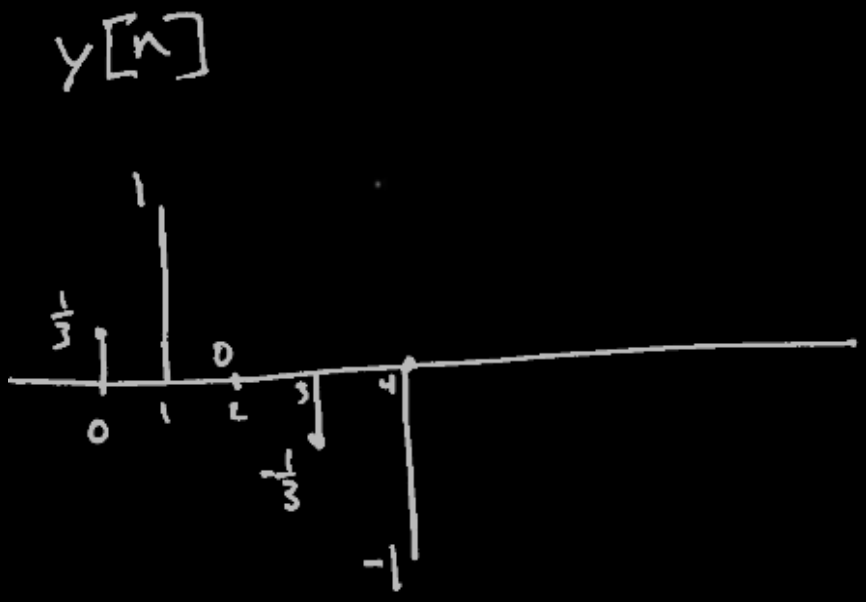
Nathan Yee
QEA B-set 3

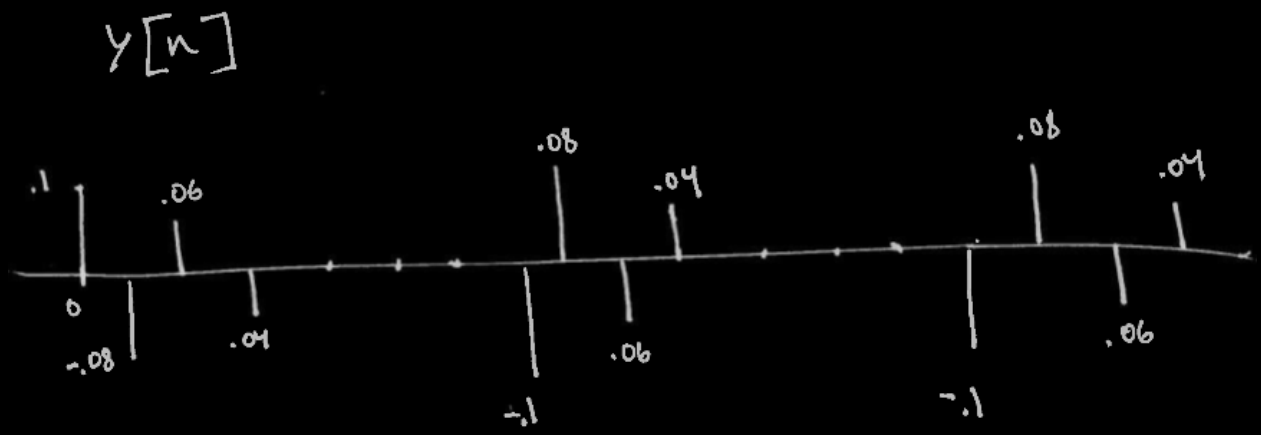
1. y[n] shown below in blue.



2. Plot of y[n] below shown in white.



3. y[n] shown below in white. I accidently forgot the .05 but that was just a simple mistake.



4.

a. A finger snapped is close to an impulse. Maybe it covers the full frequency range.
I don't really understand why the impulse or full frequency range is important. Those are really just guesses.

b.

4. Slash Guitar

In[81]:=
```
slash = Import["guitar_stairwell.mat", "Data"];
```

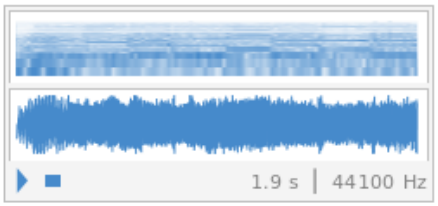
Extract .mat file into variables

In[82]:=
```
Fs = IntegerPart@slash[[1]][[1]][[1]];
h$stairwell = Flatten@Transpose@slash[[2]];
x = Flatten@Transpose@slash[[3]];
```

Play guitar sound

In[42]:=
```
ListPlay[x, SampleRate → Fs]
```

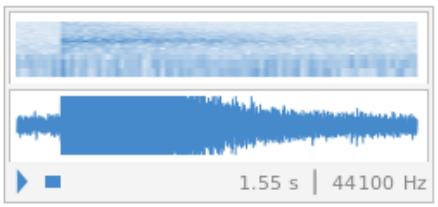
Out[42]=



Play stairwell snap

In[59]:=
```
ListPlay[h$stairwell, SampleRate → Fs]
```

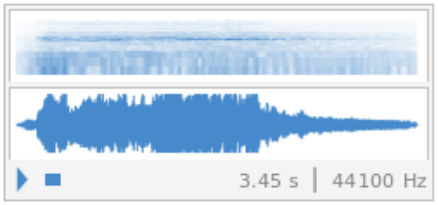
Out[59]=



Convolution with maximal overhang and zero padding

In[80]:=
```
ListPlay[ListConvolve[h$stairwell, x, {1, -1}, 0], SampleRate → Fs]
```

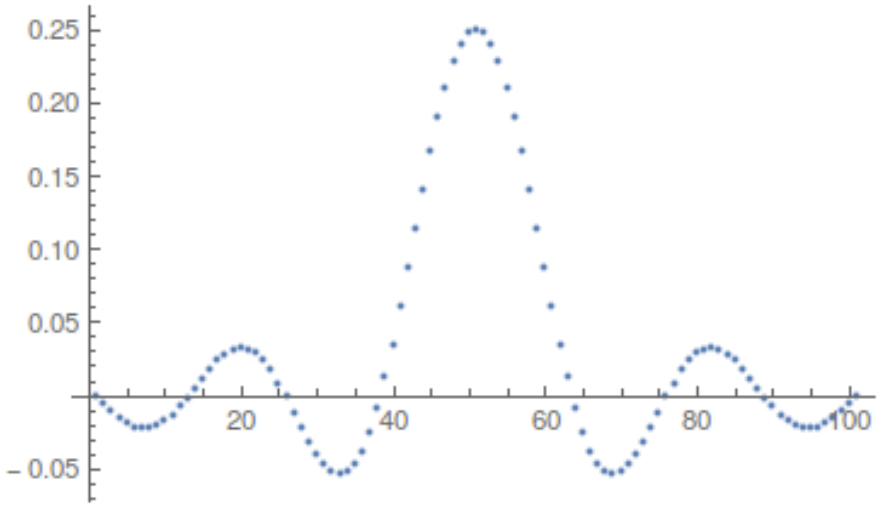
Out[80]=



5.

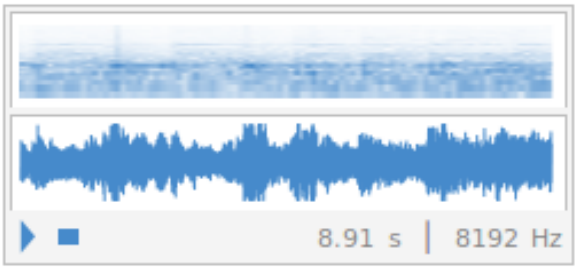
a. First make lowpass filter.

```
n = Range[-50, 50];
wc = Pi / 4;
h$lowpass = wc/Pi Sinc[(wc * n)/Pi];
ListPlot[h$lowpass]
```

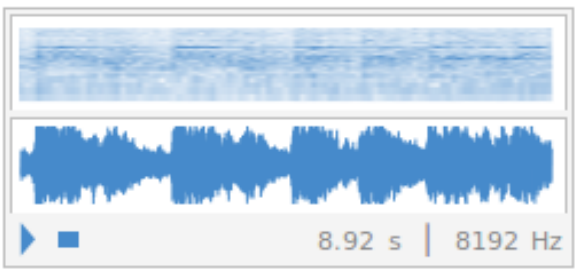


Low pass filter applied below. Sounds great.

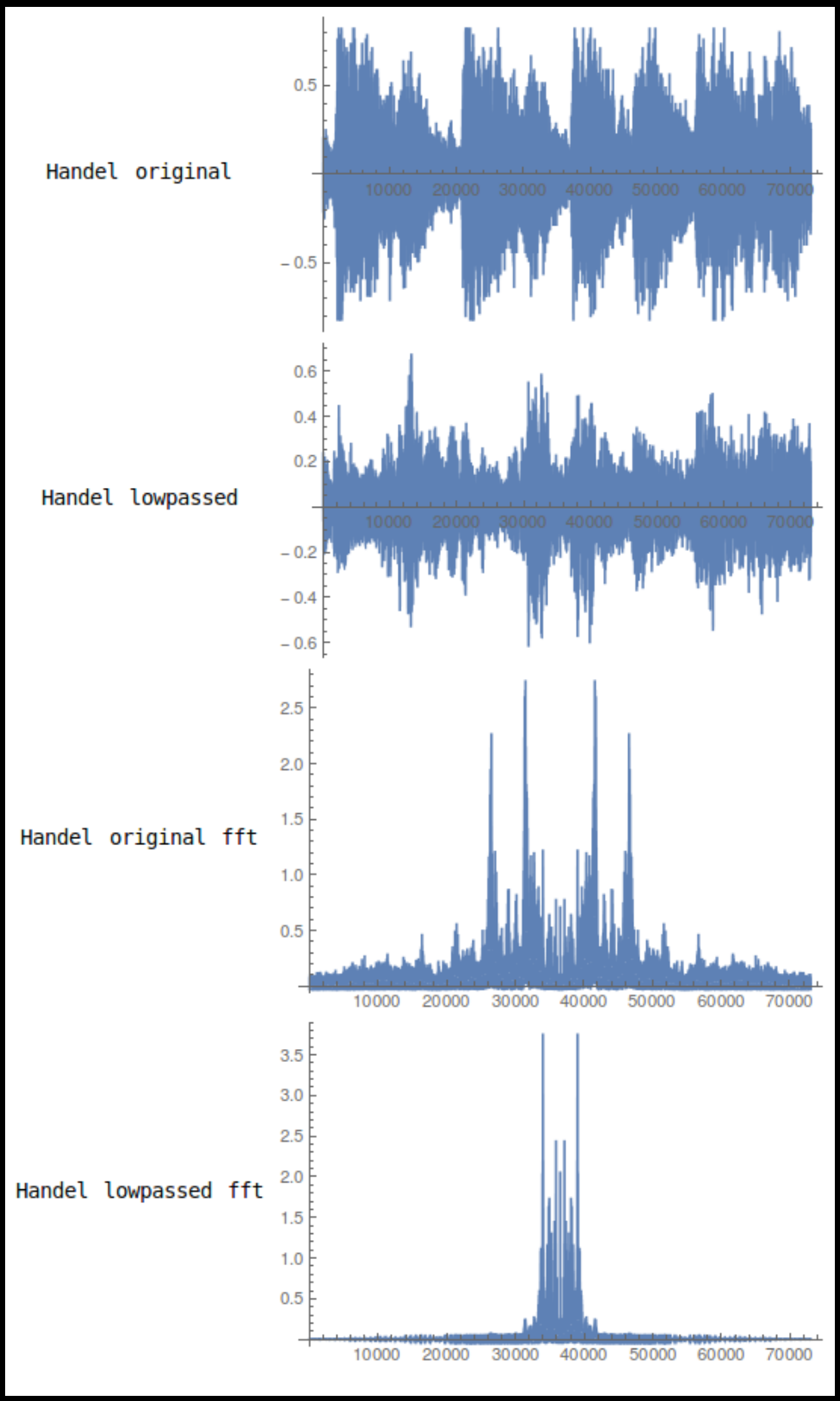
```
ListPlay[handelLowpassed, SampleRate → FsHandel]
```



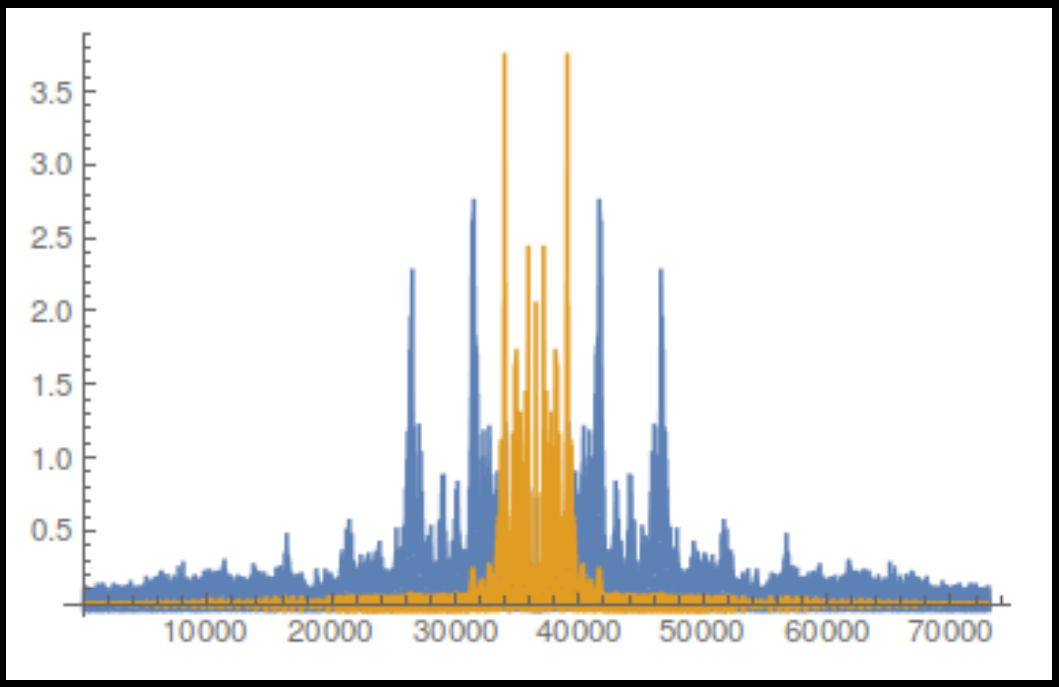

```
ListPlay[handel, SampleRate → FsHandel]
```



Plots of the original and low passed signals. Also included plots of their fft. Here I noticed that the low passed fft has lower signals than the original fft.



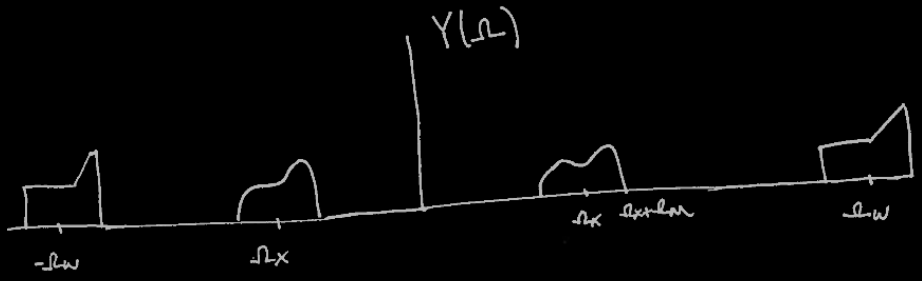
Here is the fft of the original and low passed signals. We can see that the lowpass is creating lower frequencies in addition to low passing the signal. I don't quite understand what is going on here.



b. Not done because I couldn't get low pass to work. I didn't understand what the delta was meant for. I know it was talked about earlier in the bset but I couldn't figure out what to do with it to make the highpass filter.

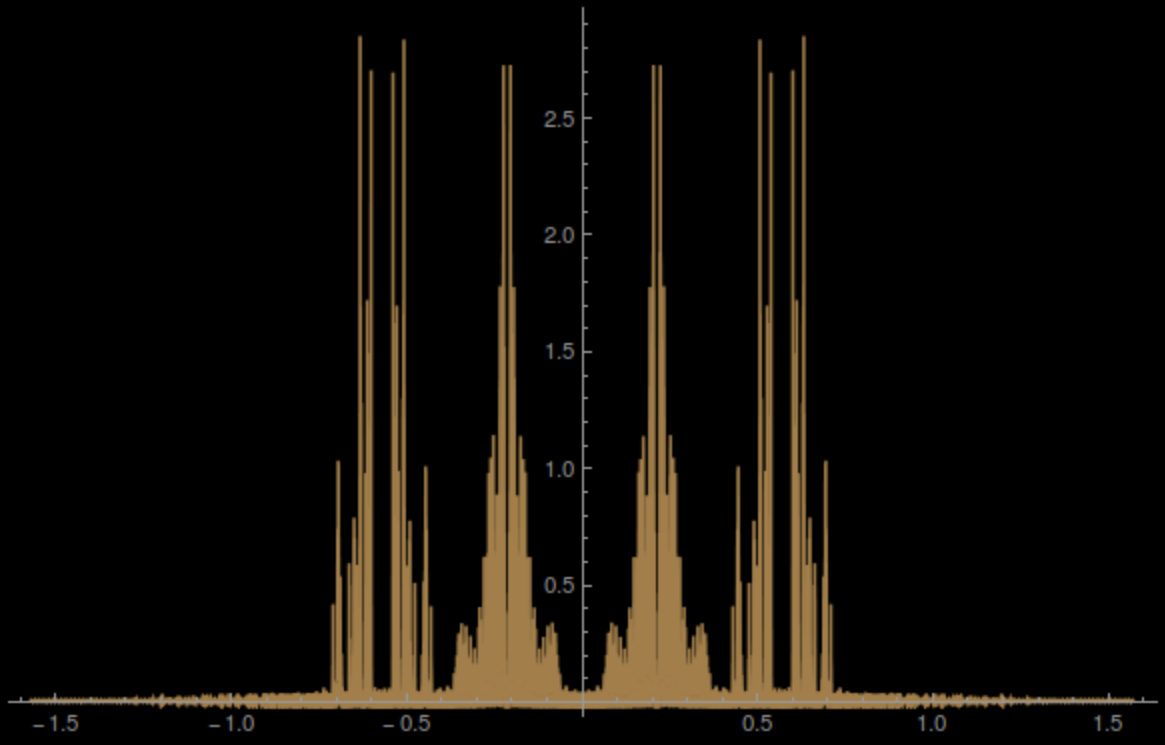
6.

a. Plot of Y(Omega) shown below. Because omega



b. To recover X(omega), we can multiply by Cos(omega), band pass the center, and multiply by two.
To recover W(omega), we can multiply by Wos(omega), band pass the center, and multiply by two.

c. Plot of fft. Notice that we can see the two distinct omegas as both signals are distinctly separated.



d. Code snipet of shifting frequencies using Cos.

Try to extract sounds by multiplying by cos

```
Ωx = (16000 / Fs$6) π;
Ωw = (6000 / Fs$6) π;
ΩM = 0.3;
```

In the case of shifting, we use MapIndex to refer to the position of the item in the list.

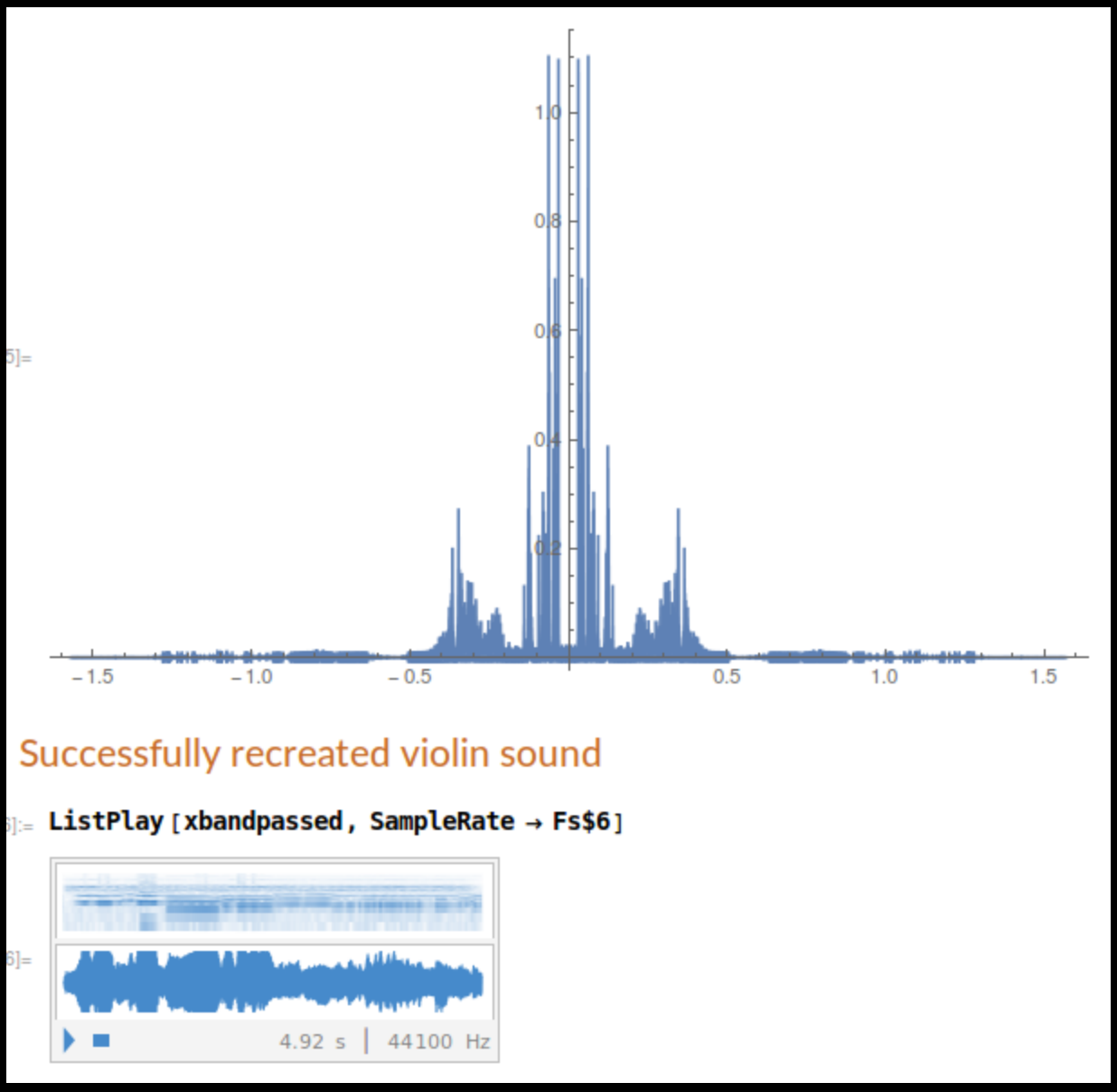
```
xshifted = Flatten @ Transpose @ MapIndexed[Cos[Ωx * #2] * #1 &, twoAMData];
wshifted = Flatten @ Transpose @ MapIndexed[Cos[Ωw * #2] * #1 &, twoAMData];
```

Second code snippet showing a bandpass filter being applied from 0 to .15 Pi. I sort of did this with trial and error. I don't really understand why these numbers worked while -.3 Pi to .3Pi breaks mathematica. I also couldn't completely remove high frequency noise.

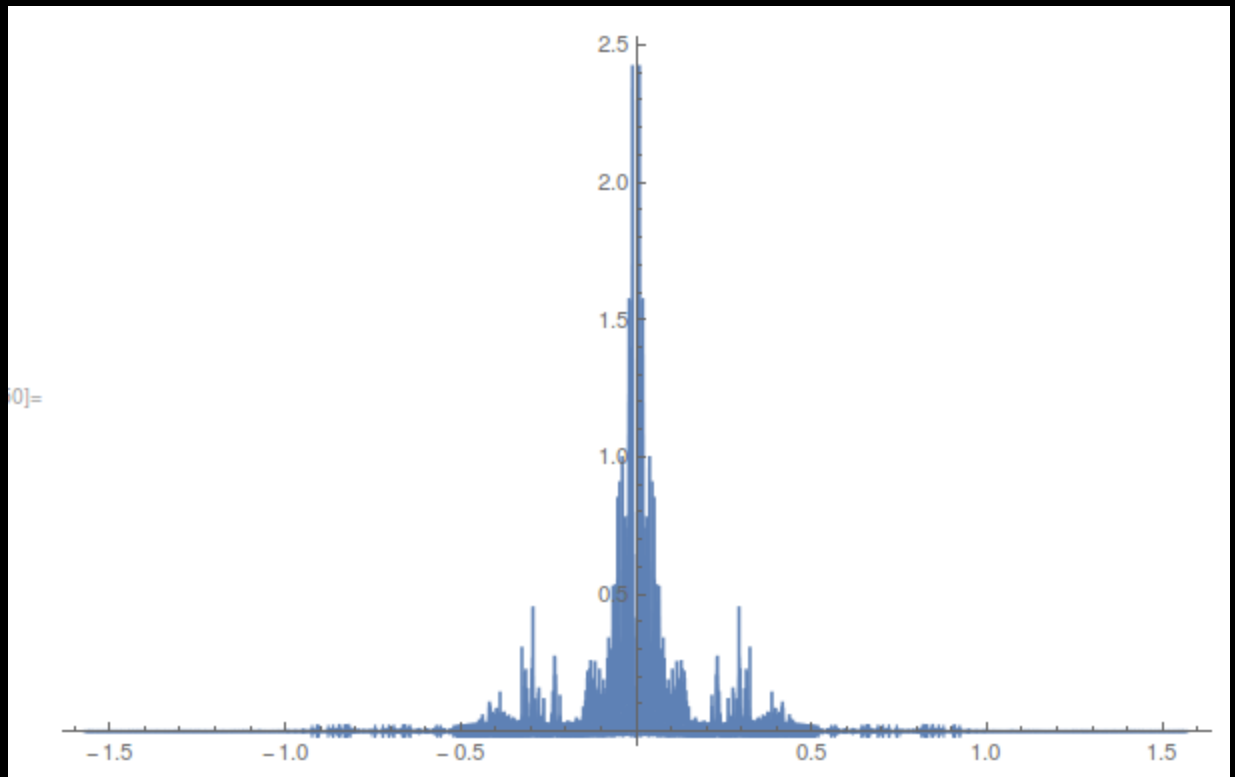
Bandpass is done on the DT shifted signal. Not on the fft.

```
xbandpassed = BandpassFilter[xshifted, {0, .15 Pi}]
```

Plot of fft and the sound file shown below.



Plot of the guitar sound below. Followed the same procedure as the violin.



Successfully recreated guitar sound

```
ListPlay[wbandpassed, SampleRate → Fs$6]
```