Notes and Reflections

As always, blue text is reflection. This bset went pretty well, I wish that I could have spent a little more time on problems 13-14, but it was a fun first-pass at FFT-based filtering. I hope we go through more examples like this so I can really internalize what's happening.

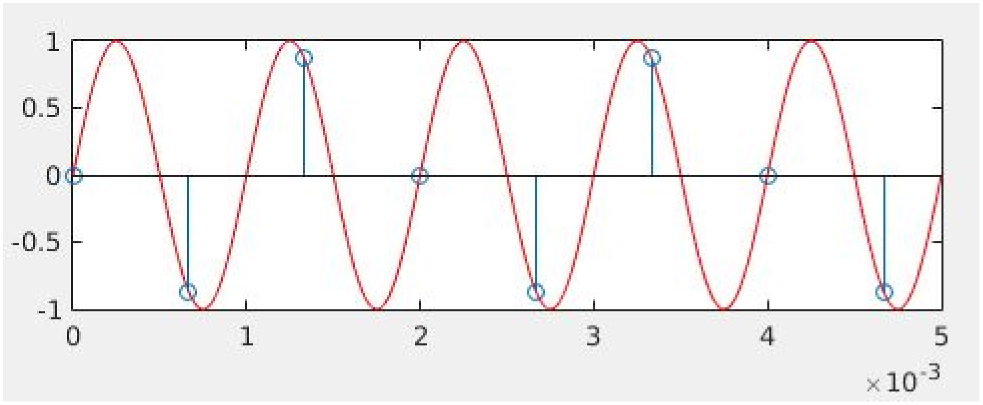
1. Sampling rates

(a) How many samples are generated per cycle of $x(t)$ when it is sampled every $1/10000$ seconds and $1/5000$ seconds?

10 samples per cycle / 5 samples per cycle

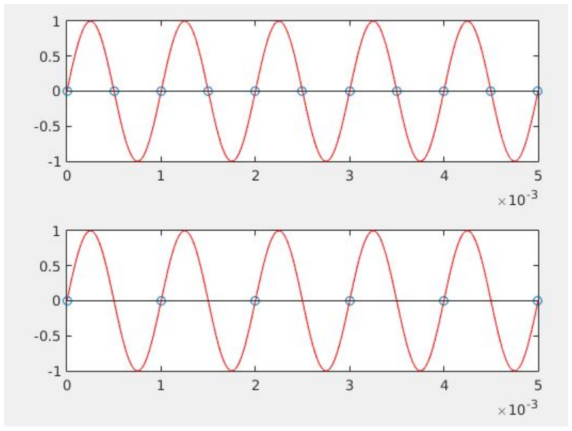
(b) What happens to the number of samples and the representation of the waveform if the sampling period is increased?

The provided code shows the samples overlaid with the original data, making it harder to tell what's happening to the reconstruction. Rebuilding from something substantially slower than half the wavelength appears impossible.

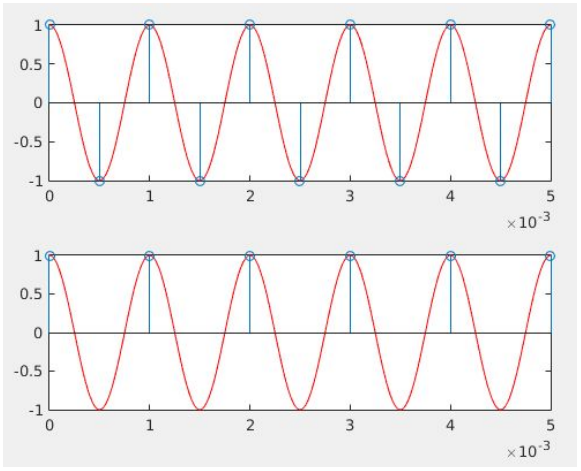


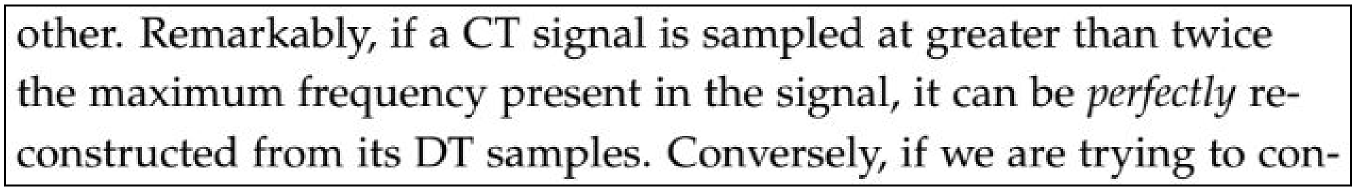
1.5 samples / wavelength

For part c), because both proposed frequencies are exact multiples of the signal frequency, all the interesting data will be lost and only amplitudes of 0 will show up.



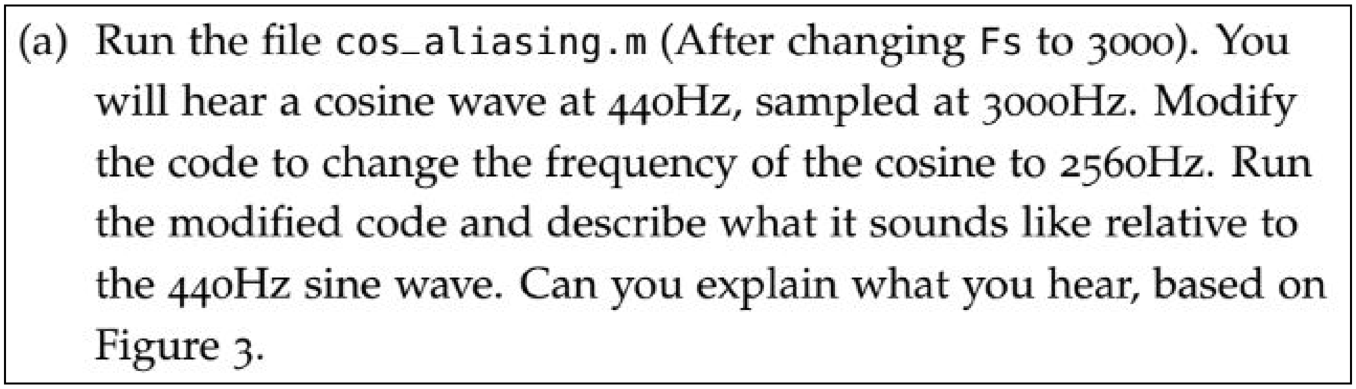
d) as expected, data is (sorta) present at 2khz, but totally lost at 1khz.




other. Remarkably, if a CT signal is sampled at greater than twice the maximum frequency present in the signal, it can be *perfectly* reconstructed from its DT samples. Conversely, if we are trying to con-

How many samples does this require? It seems intuitively impossible that you can reconstruct an arbitrarily complex signal from a finite number of points.

2. Aliasing

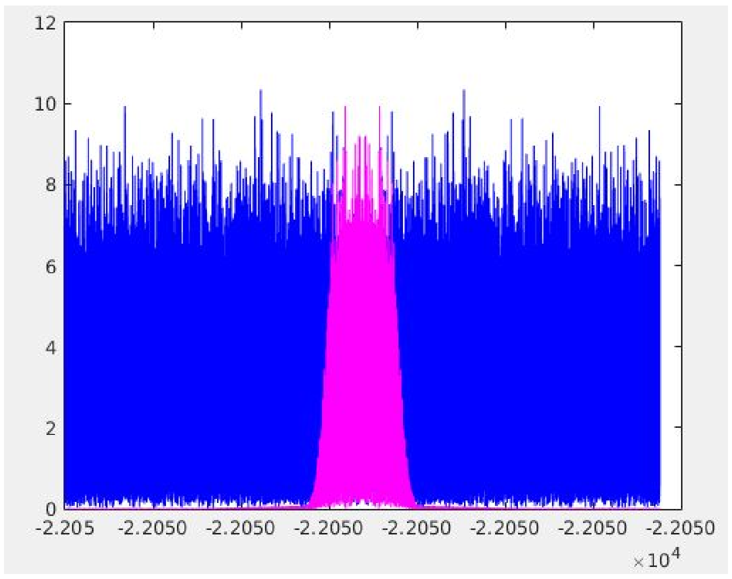

(a) Run the file `cos_aliasing.m` (After changing `Fs` to 3000). You will hear a cosine wave at 440Hz, sampled at 3000Hz. Modify the code to change the frequency of the cosine to 2560Hz. Run the modified code and describe what it sounds like relative to the 440Hz sine wave. Can you explain what you hear, based on Figure 3.

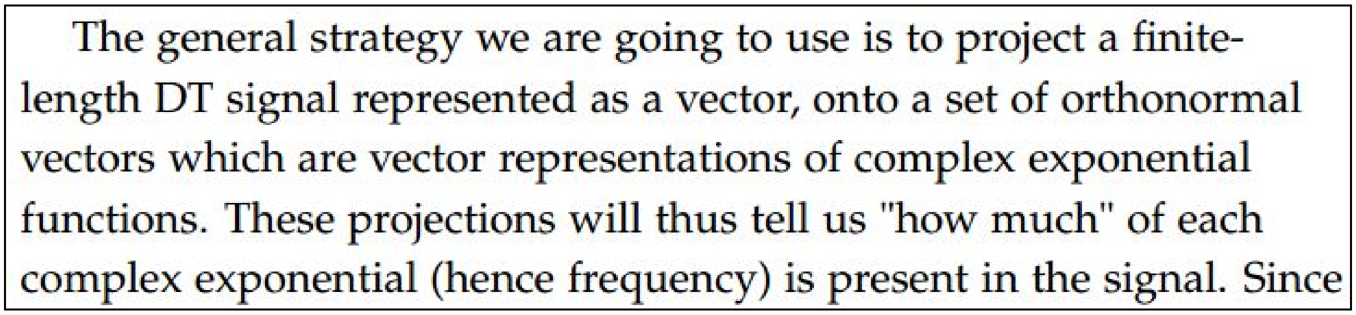
As expected, the two sounds appear identical. The 2560hz waveform is completing almost one full cycle per sample, with the interference frequency showing up at (3000-2560 hz = 440hz).

b) As expected, the higher tone is higher.

3. Qualitative frequency analysis

The “noisiness” of the apparent sound is reflected in the broad range of frequencies all overlapping each other. The pink ‘If’ sound perceptually seems lower in pitch, as reflected in the fact that it contains very few frequency components of high absolute value. Blue is white noise, pink is brown noise.




The general strategy we are going to use is to project a finite-length DT signal represented as a vector, onto a set of orthonormal vectors which are vector representations of complex exponential functions. These projections will thus tell us "how much" of each complex exponential (hence frequency) is present in the signal. Since

I interpret this to mean that we are (essentially) finding the correlation of the signal and a bunch of precomputed pure sinusoids. My concern is that this method seems very phase-dependent, although perhaps that can be avoided by testing both a sin() and cos() function at the same frequency.
Looking back (from problem 5): You tricky complex-number bastards! Sin and Cos are uncorrelated in real space, but j multiples of each other in complex space!

4. Complex #s review

(a) $Ae^{j\theta} \cdot Be^{j\phi}$

$$= (A\ B)\ e^{j(\theta+\phi)} \ = \ AB(cos(\theta+\phi) + j\ sin(\theta+\phi))$$

(b) $|e^{j\theta}|$

$$1$$

(c) $e^{j\pi}$

$$-1$$

(d) $e^{j2\pi}$

$$1$$

Note to self: Ω is in units of radians/second

5. Projections review

```
In[8]:=  {v.c1, v.c2}
```

Out[8]= $\left\{\frac{3}{\sqrt{2}} + \sqrt{2},\ -\frac{3}{\sqrt{2}} + \sqrt{2}\right\}$

```
In[12]:= Simplify[(v.c1) c1 + (v.c2) c2]

Out[12]= {2, 3}
```

6. Special case

a) Skipped
b) They form an octagon centered on the origin
c) Because they are symmetric around the origin rotationally, their sum must lie at 0

7. General case

$$S_n = \frac{a_1(1-r^n)}{1-r}$$

Finite geometric series sum to (a bunch of stuff) * (1 - the next term).
In this case, the next term is calculated when $\ell = N$, so

$$e^{\frac{j2\pi\ell(k-i)}{N}} = e^{j2\pi(k-i)} = 1$$

That means (1 - the next term) is 0, so the (bunch of stuff) doesn't matter.

I'm skeptical that something fishy is happening here, particularly driven by the assertion that $e^{2\pi i} = 1$ "if *i* is a nonzero integer". In the case where *i* is zero, this argument should still hold, as "anything to the zeroth power is one" is a mantra I've heard many times. Unfortunately, that would break problem 8.

8. Show that dot products still work

First, note that v_i is composed of a bunch of elements each having magnitude $1/\sqrt{N}$. Taking the conjugate of each of those numbers does not change its magnitude, so this statement holds for v_i^H as well. The product of a number and its complex conjugate is always a non-negative real, and because magnitudes multiply, it must have magnitude $1/N$. The dot product is defined as the sum of all N such products, so must equal 1.

9. Show that decomposition still works

First distribute out the multiplication. Because the a_i are scalars, we can use commutativity to rearrange each term to read $a_i v_i^H v_k$ Based on the previous two problems, we know that this is zero unless $i = k$, in which case it is a_i so $a_i = v_i^H x$.

10. DFT by hand

Starting with the form of the DFT in equation (18):

$$\tilde{\mathbf{x}} = \begin{bmatrix} \mathbf{v}_0^H \mathbf{x} \\ \mathbf{v}_1^H \mathbf{x} \\ \vdots \\ \mathbf{v}_{N-1}^H \mathbf{x} \end{bmatrix}$$

Observe that using the corrected simplification that $cos(\theta) = \frac{1}{2}(e^{j\theta} + e^{-j\theta})$,

$$\vec{x}_i = \frac{1}{2}(e^{j\frac{\pi}{4}(n-1)} + e^{-j\frac{\pi}{4}(n-1)})$$

Factoring out a $\sqrt{N}$

$$= \frac{\sqrt{N}}{2}(\frac{1}{\sqrt{N}} e^{j\frac{2\pi}{8}(n-1)}) + \frac{\sqrt{N}}{2}(\frac{1}{\sqrt{N}} e^{-j\frac{2\pi}{8}(n-1)})$$

The first term in this is equivalent in form to the definition of v_i given in (8) for $i = 8$, and the second term is equivalent to v_{-8}. Thus, we can write $\tilde{x}$ as

$$\vec{x} = \frac{\sqrt{N}}{2} v_8 + \frac{\sqrt{N}}{2} v_{-8}$$

This implies that for any $i \neq 8 \; and \; i \neq -8$

$$\tilde{x}_i = 0$$

because v_i is orthogonal to both $v_8 \; and \; v_{-8}$.

On the other hand, if $i = 8 \; or \; i = -8$

$$\tilde{x}_i = \frac{\sqrt{N}}{2} = 4$$

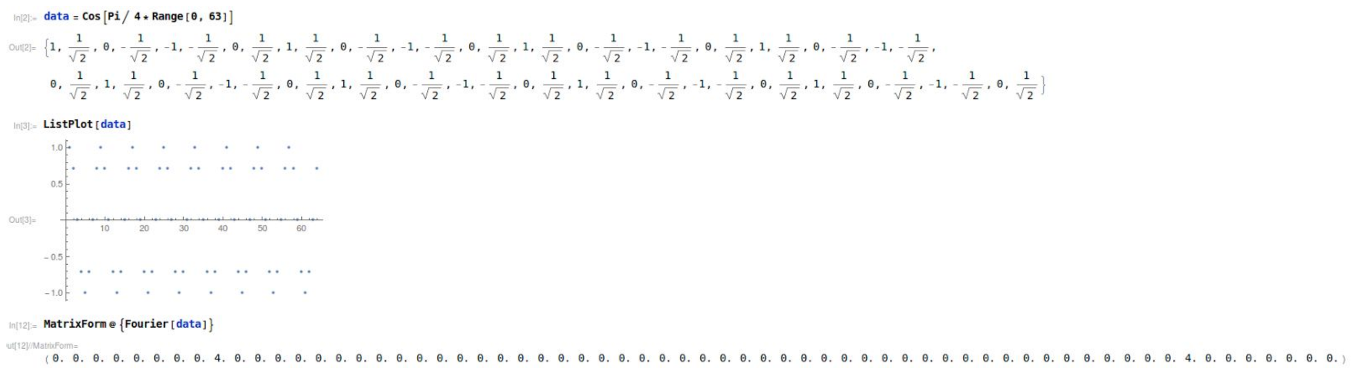
As shown in Problem 12, the DFT is periodic with length N, so $i = -8$ is equivalent to $i = 56$.

Putting these together,

$$\tilde{x} = [0, ..., 0, 4, 0...., 0, 4, 0, 0, ...\ 0]$$

Where the 4's are in the 8th and 56th positions

11. DFT by computer



```
In[2]:= data = Cos[Pi / 4 * Range[0, 63]]

Out[2]= {1, 1/√2, 0, -1/√2, -1, -1/√2, 0, 1/√2, 1, 1/√2, 0, -1/√2, -1, -1/√2, 0, 1/√2, 1, 1/√2, 0, -1/√2, -1, -1/√2, 0, 1/√2, 1, 1/√2, 0, -1/√2, -1, -1/√2,
  0, 1/√2, 1, 1/√2, 0, -1/√2, -1, -1/√2, 0, 1/√2, 1, 1/√2, 0, -1/√2, -1, -1/√2, 0, 1/√2, 1, 1/√2, 0, -1/√2, -1, -1/√2, 0, 1/√2, 1, 1/√2, 0, -1/√2, -1, -1/√2, 0, 1/√2}

In[3]:= ListPlot[data]

Out[3]=

In[12]:= MatrixForm@{Fourier[data]}

Out[12]//MatrixForm=
  (0. 0. 0. 0. 0. 0. 0. 0. 4. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 4. 0. 0. 0. 0. 0. 0. 0.)
```

12. Periodicity

(a) Show that the N-point DFT is periodic with period N, in other words, using (22), show that

$$X_{i+N} = X_i \tag{23}$$

$$X_{i+N} = \frac{1}{\sqrt{N}} \sum_{k=0}^{N-1} x[k] e^{-\frac{2\pi}{N}(i+N)k}$$

$$= \frac{1}{\sqrt{N}} \sum_{k=0}^{N-1} x[k] e^{-\frac{2\pi}{N}ik-2\pi k} = \frac{1}{\sqrt{N}} \sum_{k=0}^{N-1} x[k] e^{-\frac{2\pi}{N}ik} * e^{-2\pi k}$$

$$= \frac{1}{\sqrt{N}} \sum_{k=0}^{N-1} x[k] e^{-\frac{2\pi}{N}ik} = X_i$$

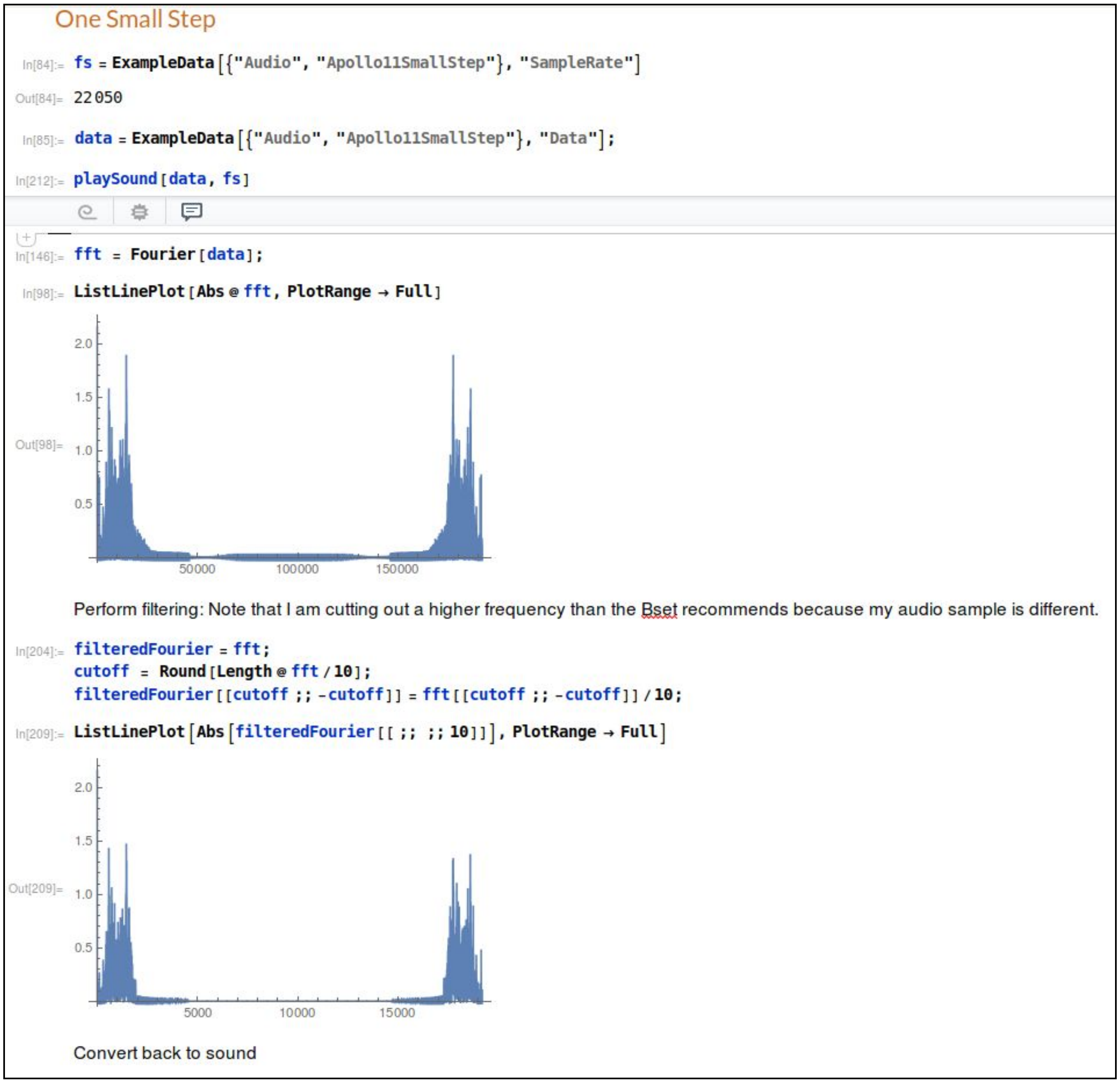
(b) Assuming that N is even, find the smallest positive value of k for which $X_k = X_{-1}$.

$$k = N - 1$$

(c) Assuming that N is even, find the smallest positive value of k for which $X_k = X_{-N/2}$.

$$k = N/2$$

13. Apollo 11



One Small Step

```
In[84]:= fs = ExampleData[{"Audio", "Apollo11SmallStep"}, "SampleRate"]
Out[84]= 22050

In[85]:= data = ExampleData[{"Audio", "Apollo11SmallStep"}, "Data"];

In[212]:= playSound[data, fs]

In[146]:= fft = Fourier[data];

In[98]:= ListLinePlot[Abs@fft, PlotRange → Full]
Out[98]=
```

Perform filtering: Note that I am cutting out a higher frequency than the Bset recommends because my audio sample is different.

```
In[204]:= filteredFourier = fft;
          cutoff = Round[Length@fft / 10];
          filteredFourier[[cutoff ;; -cutoff]] = fft[[cutoff ;; -cutoff]] / 10;

In[209]:= ListLinePlot[Abs[filteredFourier[[ ;; ;; 10]]], PlotRange → Full]
Out[209]=
```

Convert back to sound

Frequency Calculations

At 22,050hz sampling rate, the highest recordable frequency is 11,025hz, which is considerably lower than the limit of human hearing. The lowest frequency is -11,025hz.

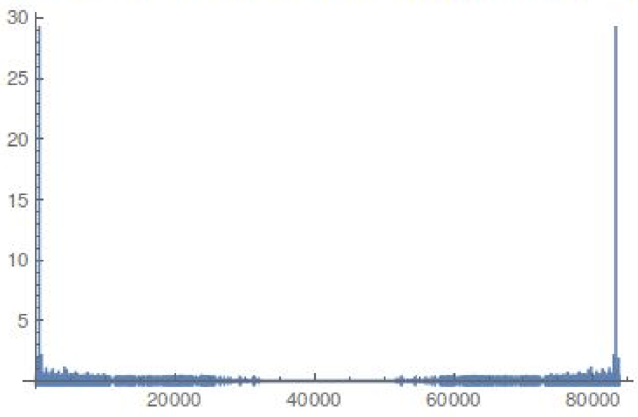
I'm still a little iffy about reading the frequencies off of these charts.

14. Guitar

Guitar

```
In[220]:= fs = Import["provided files/guitar_riff_hum.wav", "SampleRate"]
          data = Import["provided files/guitar_riff_hum.wav", "Data"];

Out[220]= 44100

In[237]:= playSound[data, fs]

In[223]:= fft = Fourier[data];

In[224]:= Rasterize@ListLinePlot[Abs@fft, PlotRange → Full]
```

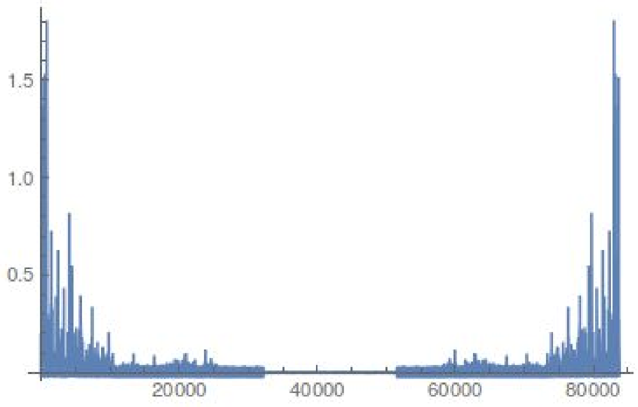
Out[224]=



Perform filtering: Looking at the graph above, I decided to neuter all overpowered freqs.

```
badFreq = Max

In[246]:= cutoff = 3;
          filteredFourier = Map[Function[val, If[Abs@val > cutoff, 0, val]], fft];

In[250]:= Rasterize@ListLinePlot[Abs[filteredFourier], PlotRange → Full]
```

Out[250]=



Convert back to sound

```
In[248]:= data2 = Re@InverseFourier[filteredFourier];
```