PHY324: Removing Noise from Data via FFT Filters

By: Gurmanjot Singh

Due: 20 January 2023

1 Introduction

The Fourier transform (FT) is a very important mathematical tool in signal processing, as it lets us decompose a signal into its fundamental oscillation frequencies. In particular, given a signal in time $f(t)$ the Fourier transform (in complex phase space) is:

$$F(\omega) = \int_{-\infty}^{\infty} f(t)e^{-i\omega t}dt$$

This gives us a different and potentially more useful representation of the signal with a simple transform. In practice, we use the Fast Fourier Transform (FFT) as the actual computer algorithm to compute this on a data set (the discrete version of FT), so we can look at the dominant frequencies that construct the wave. However, in experimental data, there usually is information in the signal from other undesired sources in the environment that result in the frequency spectrum being filled with a small fuzz of noise. It is of interest in a lot of fields to attempt to filter this noise out, as it distorts the original signal and makes analysis more difficult.

To do this, we will apply a filter function $w(\omega)$, basically redistributing the weights of the frequencies when multiplied with the FFT of our signal so that the noise spectrum nearly vanishes and only the non-negligible peaks remain. This gives us a modified FFT $F^* = w \cdot F$, which then we take the inverse FFT of to recover the original signal with some of the noise removed. This is done usually by setting the filter function to be the sum of Gaussians centered at each individual peak, with a chosen width based on how certain we are of the peak's location.

We attempt to study this process exactly in this report, on some given position-time data of a signal.

2 Methods

In this report, we will first verify what the FT does, by observing how the FFT acts on the superposition of two sine waves of different amplitude and frequency that we have knowledge of, checking if it lets us recover the original amplitudes and frequencies used.

After this, we will apply this understanding on the process of recovering the true signal out of data cluttered with noise, by attempting to filter out the frequencies in the Fourier transform that are not part of the signal's profile, which we are given, using the technique outlined in the introduction. We then compare this to the true signal to determine the accuracy of this method. Note that we add the noise artificially beforehand, which is Gaussian distributed around 0 with a standard deviation of half the amplitude of the pure wave, so as to not overpower the actual wave's behaviour.

With our new understanding, we use a similar procedure on much more noisy data without access to the true signal, so we do not have a reference that we can use to verify if our filtration was accurate. This resembles how noise filtering algorithms have to decide what parts of a signal to remove as noise based on its profile, usually by deciding only the significant peaks in the graph are part of the real signal. As humans, it is decidedly easier for us to do this using our eyes.

Finally, we consider a Fourier decomposition of a signal with time varying frequency and attempt to interpret the meaning of such a decomposition, since the Fourier transform does not really map to a time dependent frequency domain.

All of this will be done with Python code that evenly samples 200 points ifor the first two signals, 2000 points for the third signal, and 1000 points for the last signal, all in 1 second of unit time. The results will be plotted and discussed in the next section.

3 Results

We will first look at the FFT of the following pure position signal:

$$f(t) = A_1 \sin(\frac{2\pi}{t_1}(200t)) + A_2 \sin(\frac{2\pi}{t_2}(200t))$$

for $A_1 = 5\text{m}$, $A_2 = 9\text{m}$, $t_1 = 17\text{s}$, and $t_2 = 13\text{s}$. The graph of this signal is given by the following figure:

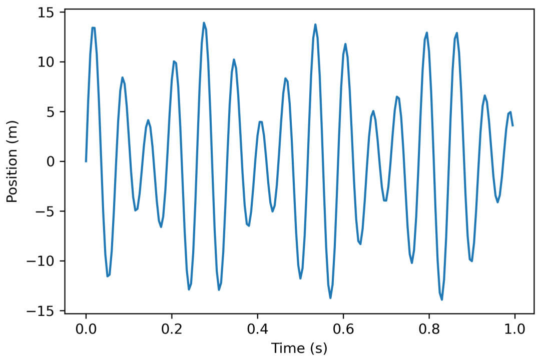

Figure 1: Graph of sampled f.

We perform the FFT to convert to the amplitude-frequency domain. Note that we scale down the amplitude axis by 200, since the FFT does not take the average over the frequencies as in the regular Fourier transform and so does not preserve units. We will also take the absolute value to remain on the real axis, as we only really need the amplitude and frequency location, not the phase. This gives us a graph:

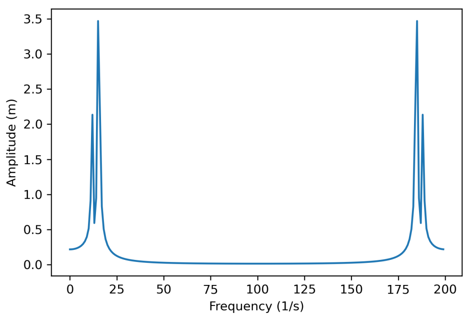

Figure 2: Graph of absolute value of FFT of sampled f.

Now, as an approximation of the true Fourier transform, the FFT decomposes a sine wave $\sin(2\pi ft)$ of frequency ρ into two primary frequency peaks in its graph, given by one at ρ and another at $-\rho$, modulo the sampling size (this is an artifact of the math behind the FFT). Moreover, these are scaled by half the true amplitude since $\sin(x) = 1/2e^{ix} - 1/2e^{-ix}$.

We observe this in this graph, as the two symmetric peaks of the first signal and second signal respectively about 100, of which the first two peaks' location can be used to extract the frequencies of the sine waves $f_1 = 200i/t_1$ and $f_2 = 200/t_2$. The height of these peaks then gives us the amplitudes via the normalizing equation $H_1 = A_1/2$ and $H_2 = A_2/2$.

We can zoom in on the peaks in an attempt to observe these:

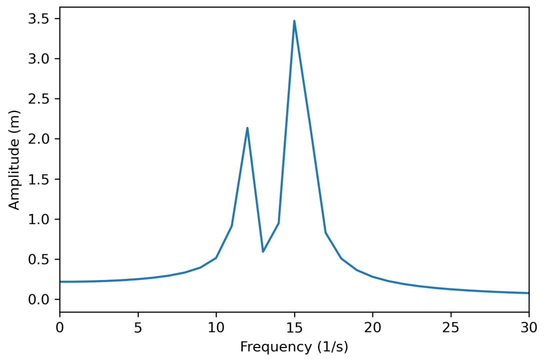

Figure 3: Graph of absolute value of FFT of sampled f, $0 \leq f \leq 30\text{s}^{-1}$.

By eyeballing it, we get that $f_1 \approx 12\text{s}^{-1}$, $f_2 \approx 15\text{s}^{-1}$, $A_1 \approx 4.5\text{m}$ and $A_2 =\approx 7\text{m}$. Given our actual t_1 and t_2 values, we know the real frequencies are $f_1 = 11.76\text{s}^{-1}$ and $f_2 = 15.38\text{s}^{-1}$ up to the second digit, while the real amplitudes are $A_1 = 5\text{m}$ and $A_2 = 9\text{m}$.

This gives us an error range of approximately $\pm 0.5\text{s}^{-1}$ on the frequencies, and one of about $\pm 2\text{m}$ on the amplitudes, which is pretty big, but considering the FFT is only an approximate Fourier transform, it is possible that information is lost due to the limited sampling size, which appears in how the peaks aren't perfect delta distributions, but are smeared across an interval of $\pm 1\text{s}^{-1}$.

Regardless, this gives us a powerful tool to extract the frequencies and respective amplitudes from a signal, as long as they are large enough for the uncertainties to not play a significant role in anything but the noise.

Using this, we consider a simpler position-time signal given by:

$$g(t) = A_1 \sin(\frac{2\pi}{t_1}(200t))$$

as before, but now perturbed by random Gaussian noise that has a standard deviation of half the amplitude. Since the error in the amplitude is much smaller, it should be easy to recover the true signal's frequency-amplitude profile using a filter. In particular, our original signal and noisy signal are given by the following graphs: We then apply the filter function given by two Gaussians at

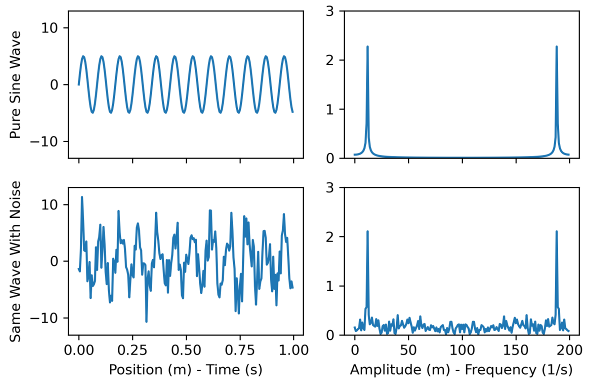

Figure 4: Graph of sampled g and absolute value FFT with and without noise.

the symmetric peaks, which is of the form:

$$L(f) = e^{-(f-f_0)^2/l} + e^{-(f+f_0-200)^2/l}$$

for the base frequency f_0 that we align sufficiently with the actual peaks, and a width l based on how precise we want our filter. The modified FFT will be of the form $L(f) \cdot \mathrm{FFT}(g)(f)$ and will effectively zero out the noise, which we can then invert back to our original wave up to some error.

Note that without hindsight, this process simply requires us to look at the graph and guess the frequency and amplitude values, for which I would estimate $f_0 \approx 12\mathrm{s}^{-1}$, while setting a width of $l = 0.5\mathrm{s}^{-2}$, since even without any noise, the Fourier transform is smeared due to the sampling limit. These choices give us the following filtration result: (next page)

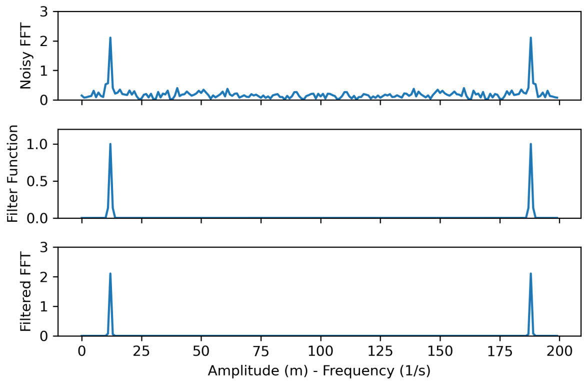

Figure 5: Graph of the absolute value of FFT of g with noise and after filtering, along with filter function.

If we apply the inverse FFT, we get the following result comparison:

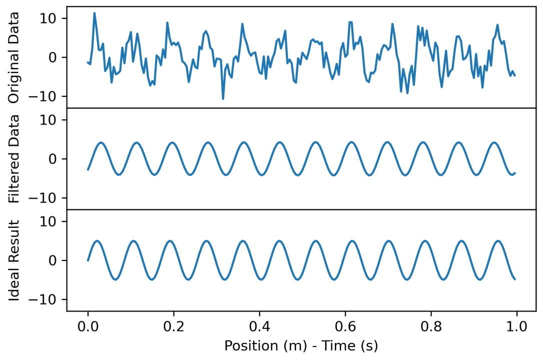

Figure 6: Graph of sampled g with and without noise, including filtered g.

We see that the filtered result is very close to the exact values of g, just slightly right-shifted by a very small phase and scaled down by a factor very close to 1 due to the error in the approximation. Note that this value was achieved by trial and error to check what estimate gave us the highest constructive interference.

Given the benefit of hindsight, we could simply just set the frequency $f_0 = 200/t_1 \approx 11.73$, which would be much more precise and would require no trial and error. However, in practice, we usually don't have the original signal, and it is our goal to attempt to retrieve the real one out of the noise as we did here.

With the methods that we have used previously, we can now attempt to clean the noise from a noisy signal without any knowledge of the original. In this case, we have a position-time signal S given by the following graph:

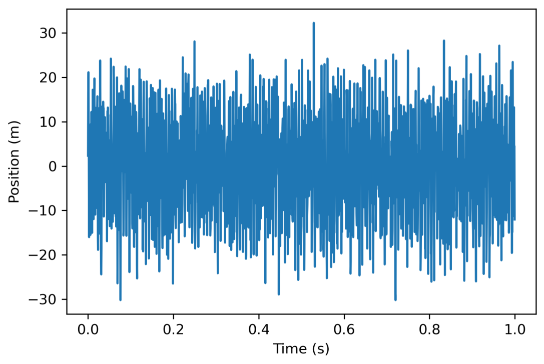

Figure 7: Graph of S without filtering.

To get an understanding of the more local behaviour, we can show just the first 300 points:

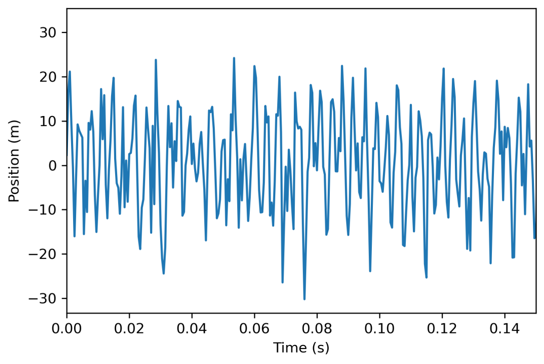

Figure 8: Graph of S without filtering, only first 300 points.

We can then transform this into the amplitude-frequency domain via FFT and appropriate rescaling to maintain the units. Note that the first 300 points of the FFT are the only relevant ones, as all the unique peaks up to symmetry arise before $300s^{-1}$, for which we can directly observe the amplitude and frequency in the way we did previously. This gives us the following: (next page)

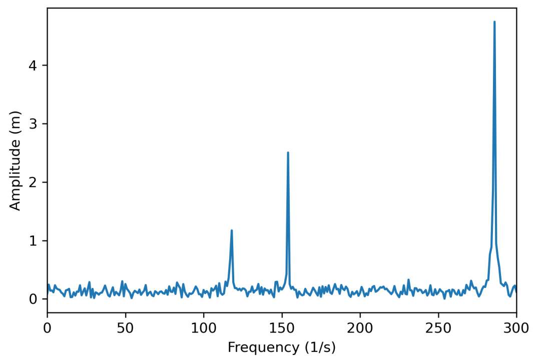

Figure 9: Graph of absolute value of FFT of S without filtering, only first 300 points.

We can make out primarily three frequency peaks, which with trial and error I found have filters that conserve the peaks almost perfectly at approximate frequencies $f_1 \approx 118\text{s}^{-1}$, $f_2 \approx 154\text{s}^{-1}$, and $f_3 \approx 286\text{s}^{-1}$, which gives us a similarly defined Gaussian filter function as a sum of the individual Gaussians at these frequencies, up to the symmetry peaks. Setting the width to be 1s^{-2} gave the most accurate matching of the peaks, suggesting that it accurately reflects the real peaks.

The graph of the filtration process is as below, for the first 300 points:

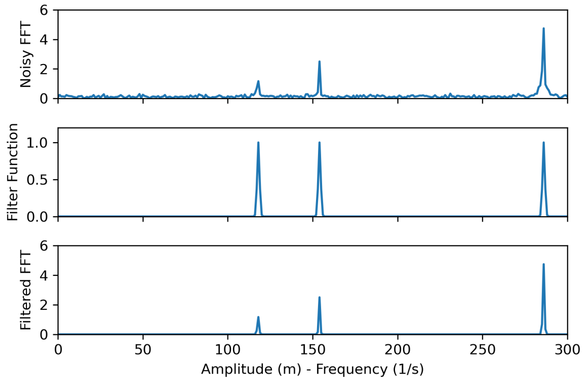

Figure 10: Graph of absolute value of FFT of S with and without filtering, as well as filter function (first 300 points).

Looking at the graph, we can approximate the amplitudes as well for each frequency, as $A_1 \approx 2.4$m, $A_2 \approx 5.0$m, and $A_3 \approx 9.0$m. This gives us a referential 'ideal' signal:

$$g^*(t) = A_1 \sin(2\pi f_1 t) + A_2 \sin(2\pi f_2 t) + A_2 \sin(2\pi f_2 t)$$

We finally transform back to the original signal's domain, which gives us the following comparison for the first 300 points:

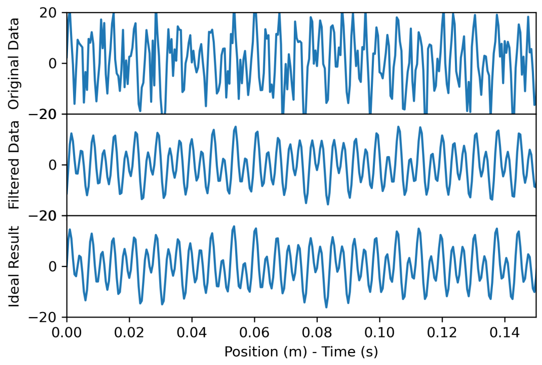

Figure 11: Graph of S with and without noise, including ideal result (only first 300 points).

Observe that although the new wave still looks like it is a bit noisy, it is nearly the same as the wave produced by three perfect sine waves at the given amplitudes (A_1, A_2, A_3) and frequencies (f_1, f_2, f_3), up to a very small phase shift and scale due to conversion errors. This noisy look is in fact due to the signal having such high frequency sine waves composing it with huge variation, so the small time scale does not reflect the periodicity very well (can still be seen).

Note that the noise in this data looks much more significant as it adds to the actual signal by an amplitude factor of around 7, shown by the maximum value of the wave. Even then, the FFT seems to very accurately visualize the location of the true frequencies in spite of the noise, since the frequency domain has the major frequencies at all the peaks.

This lets us conclude that our source signal has a true signal composed of three sine waves at the provided amplitudes and frequencies, paired in order.

We now consider how the Fourier transform and its partner, the FFT, behaves on a function with an unclear standing wave decomposition–in other words, a signal that is aperiodic, or changes its frequency over time. In our example, we use the following position-time signal:

$$h(t) = \sin(2\pi t(30t + 1))$$

I chose the frequency to vary as $30t + 1$, for an observable change in the frequency. The resulting graph, sampling 1000 evenly distributed points in from $t = 0$s to $t = 1$s, is:

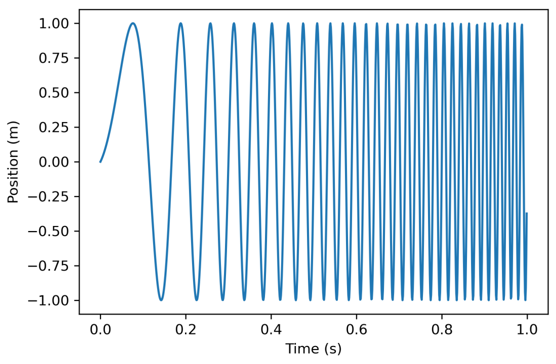

Figure 12: Graph of h.

which has one side of the symmetric-peak FFT given by:

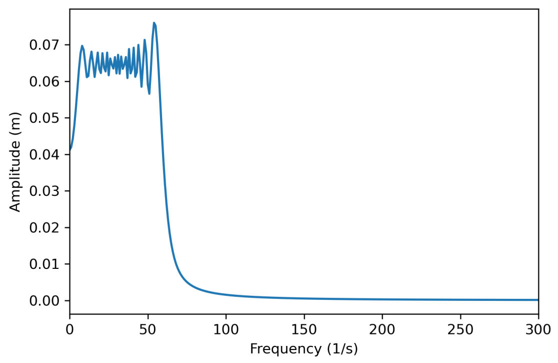

Figure 13: Graph of absolute value of FFT of h (first 300 points).

With our understanding of the Fourier transform, the FFT shows us that h has significant contributions to its signal from frequencies all the way from 10s^{-1} up to a frequency of around 50s^{-1}, distributed around 30s^{-1} on the graph.

This behaviour makes sense, since the frequency varies between 1s^{-1} and 31s^{-1}, and as the low frequency modes oscillate for less time than higher frequencies, contributions from higher frequencies close to the central frequency are more significant. The existence of frequencies contributing higher than the maximum comes from the fact that the frequency is constantly increasing, and so require accumulating contributions from some higher frequencies, although not too high as to destructively interfere and act as noise (only near the rate of increase in the frequency).

Also note that, due to a continuous distribution of non-negligible frequencies, the height of the frequency peaks are very low, at around order 0.07m. This corresponds to the fact that we sum over this distribution to retrieve our original signal, which clearly is only possible if the sum itself is relatively small and bounded. Note that the FFT picture is just an approximation of the real Fourier transform, as to why it looks so strange.

If we were to increase the time interval, the interval of allowable frequencies would increase, as the frequency would keep increasing up to the end of the interval, and so provide contributions.

4 Conclusion

In summary, we developed methods with which we could analyze a signal distorted by noise and extract the real signal out of it. In particular, we first verified the utility of the FFT as a tool for the frequency analysis of a position-time signal, by recovering a good approximation w.r.t. the noise for the frequencies and amplitudes of sine waves that composed an already known signal.

After this, we considered performing a FFT of an already noisy signal–for which the true signal was known–on which we applied a filter to remove the noise, thereby verifying that we could retrieve the (almost) pure signal by an inverse FFT and comparing the accuracy.

Once the validity of this processing technique was checked, we used it on a signal of unknown composition, which we analyzed and pulled out a sum of three distinct sine waves quite quickly, just after checking frequencies with the filter by trial and error.

Finally, we tested the limits of the FFT and of this analysis on signals with time varying frequencies, for which we found that it still adequately describes the behaviour of the function as a continuum of superposed frequencies as opposed to a few discrete peaks. In the continuum limit, this is motivation for the Fourier transform of continuous bounded non-periodic functions, and so a sort of generalized spectral decomposition of such functions into waves.

Altogether, this paper was able to verify the strength of the filter function method in extracting signals, as well as making extensions in the understanding of how the FFT and Fourier transform behave.

It is important to acknowledge, however, that the FFT is just an approximation and so introduces errors in the Fourier transform calculation, as we found with the smeared peaks and the wobbly FFT of our quadratic sine. This could cause problems when dealing with other uncertainties or noise of similar proportions if it makes up a high percentage of the relative error. Thus, this method requires more careful analysis and more precise sampling to be able to use it in other fields, albeit very useful.