PHY324 - Basic FFT based Signal Processing

Chaitanya Kumar Mahajan

January 21, 2023

1 Introduction

Fast Fourier Transform is an efficient algorithm to compute the Fourier transform of any given signal. FFT of a noisy signal can easily provide the dominant frequencies and the relative amplitudes of the peaks which can be used to rebuild the original signal without the noise.

The Fourier transform is a function transform going from time space to frequency space given by,

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

where $y(t)$ is the original time signal of frequency ω. Notice that the Fourier transform gives a complex function, and thus we generally plot the norm of function. The Fourier transform then gives us a way to calculate the frequencies in a given signal.

We also have an inverse Fourier transform which takes a frequency function and gives a time signal back.

$$y(\omega) = \frac{1}{2\pi} \int_{-\infty}^{\infty} Y(\omega) e^{i\omega t} d\omega$$

In practice we rely on a computer to calculate these transforms and thus we sum over a discrete number of points rather than an integral over all numbers.

In this report, we will demonstrate the process using a signal of two known frequencies and imposing it with random noise. We then use FFT to get the dominant frequencies and rebuild the original signal by filtering out the noise and using inverse FFT. Next we repeat the process with an unknown signal and construct a signal without major noise. And finally we consider an example where the frequency is non-constant and changes linearly with time to demonstrate that the Fourier transform is valid for even any periodic function.

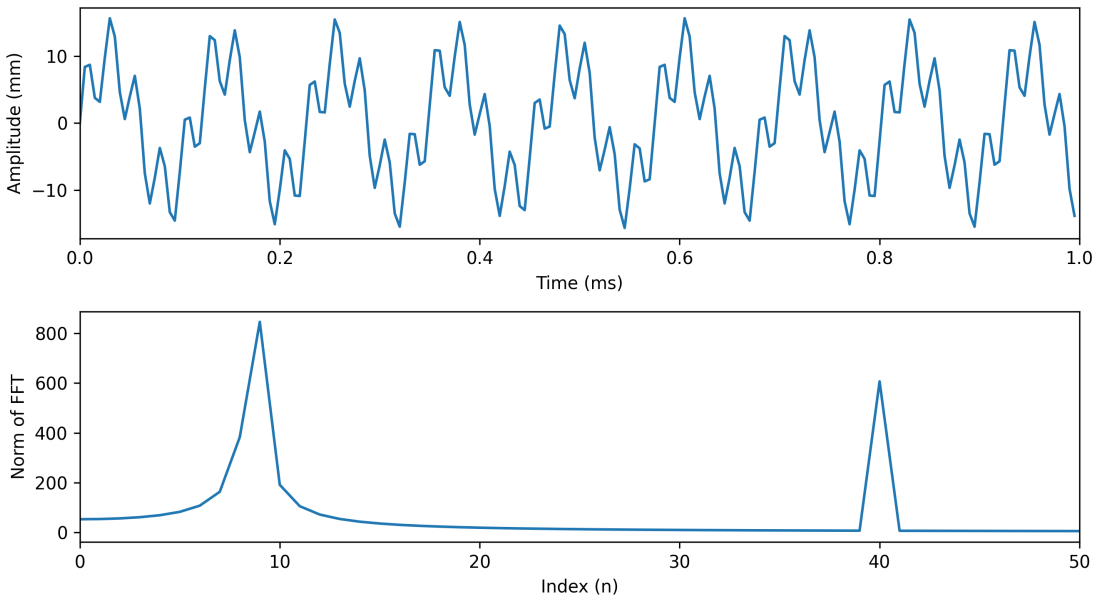

Figure 1: (Top): Plot of the original signal $y_1 + y_2$ along with, (Bottom): Norm of the FFT of the signal. We only show part of the FFT plot, as the other half is a symmetric copy and contains no extra information.

2 Known Signal

Consider the two sine waves given by,

$$y_1 = 6 \sin\left(\frac{2\pi}{5} t\right)$$
$$y_2 = 10 \sin\left(\frac{2\pi}{23} t\right)$$

where time is in (ms) and amplitude is in (mm). Notice that the time period of y_1 is 5 ms and for y_2 is 23 ms.

Adding y_1 and y_2 we generate the following plot (1T). Taking the FFT of the signal and plotting the norm of the FFT (1B) we notice that the peaks are located at 9 and 40. Since the FFT is a discrete algorithm, it indexes over the interval of the signal. In this plot, we used 200 time-steps and so the time periods are,

$$t_1 = \frac{200}{9} \approx 22.2 \pm 1.2\text{ms}, \quad t_2 = \frac{200}{40} = 5.0 \pm 0.6\text{ms}$$

which are the time-periods we used up to the uncertainty. The uncertainty is found by half width of the area around the peak.

Now since the FFT algorithm in Python outputs an array of complex numbers, we can sort through the norms of the values to find the dominant peaks. This gives us,

$$A_1 = 845 \pm 1, \quad A_2 = 607 \pm 1$$

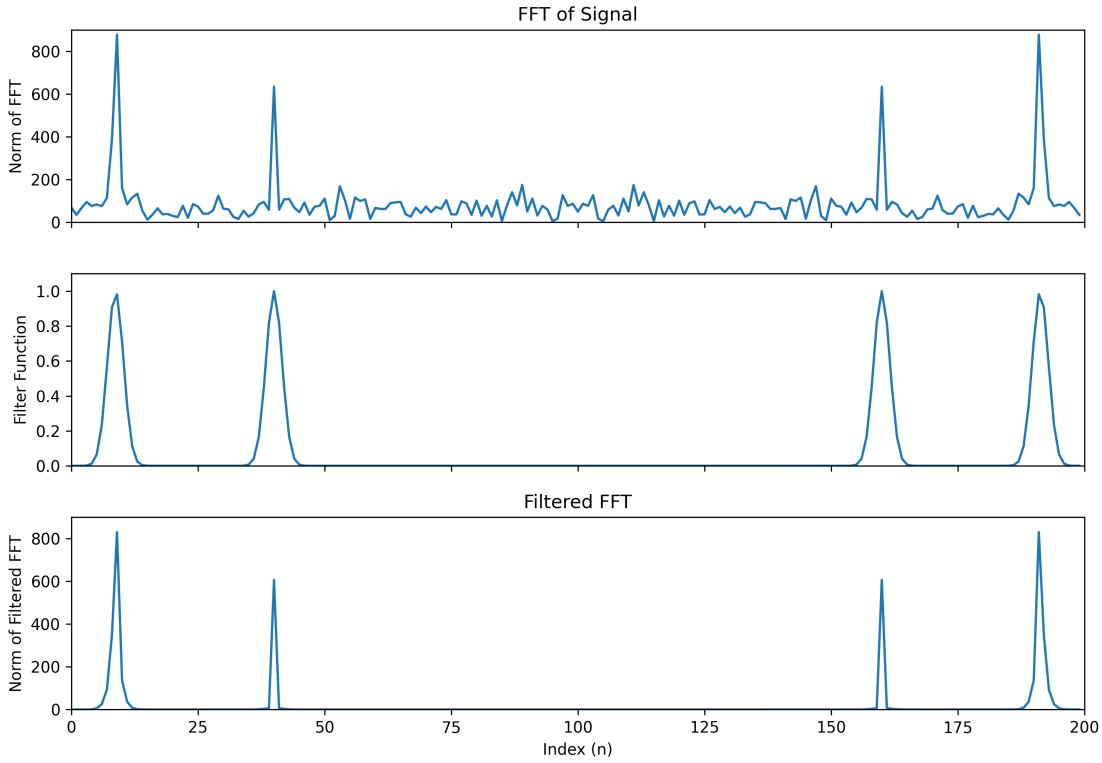

Figure 2: (Top): Fourier transform of the noisy signal and the (Middle): Gaussian Filter function used to remove most of the noise and the (Bottom): Filtered Fourier transform of the signal.

The uncertainties is calculated by half the difference in two adjacent values at each peak. From which we can calculate a rough estimate of the relative amplitudes.

$$\frac{A_2}{A_1} \approx 0.71 \pm 0.01$$

Notice that we can get an approximate relative amplitude (calculated 0.71 compared to actual 0.60) of the sine waves we used to generate the signal, however finding exact amplitudes is too complex as it requires integrating over a small range around the peak.

Now we add noise to our example to demonstrate the usefulness of a Fourier transform. We add noise to our signal by sampling from a Gaussian distribution and adding it to $y_1 + y_2$. This generate (3T) and its Fourier transform is seen in (2T).

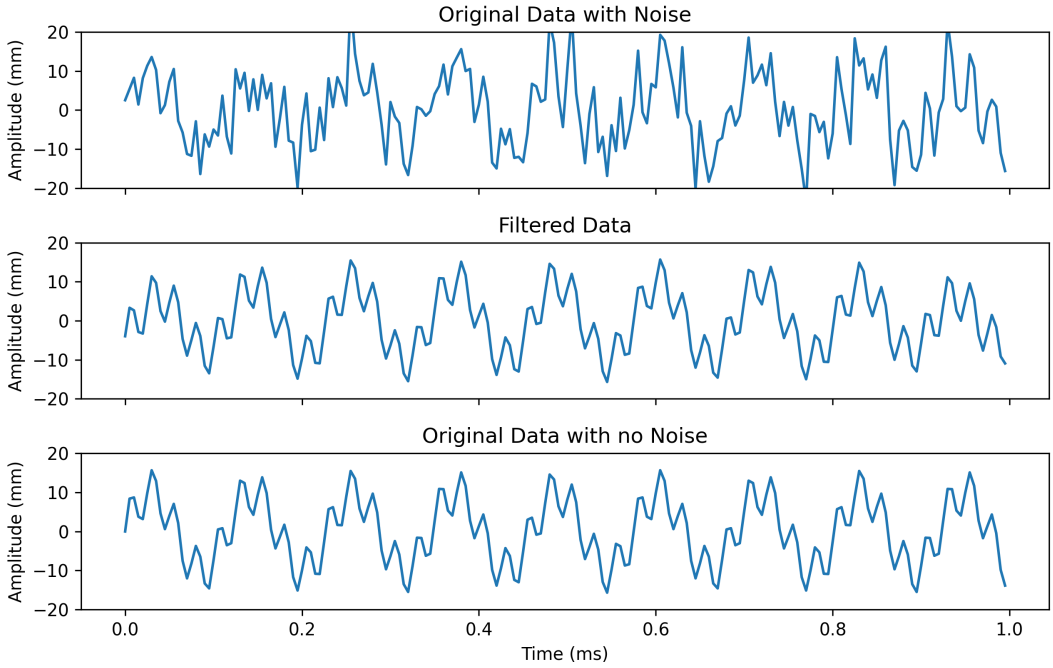

Figure 3: (Top): Plot of $y_1 + y_2$ with noise along with, (Middle): The filtered signal using a Gaussian filter function defined above, and (Bottom): The original signal $y_1 + y_2$ with no noise.

We use a filter function with two Gaussian curves centered at the index where the FFT had dominant peak, and we ensure that the width of the Gaussian curves are slightly larger that the width of FFT to include all of the signal (2M). The effect of multiplying the filter function and the noisy data is getting a Fourier transform where majority of the noise has been removed (2B). Taking the inverse Fourier transform we see that the filtered signal and the original signal are extremely similar with slight variance at the ends (3M, B).

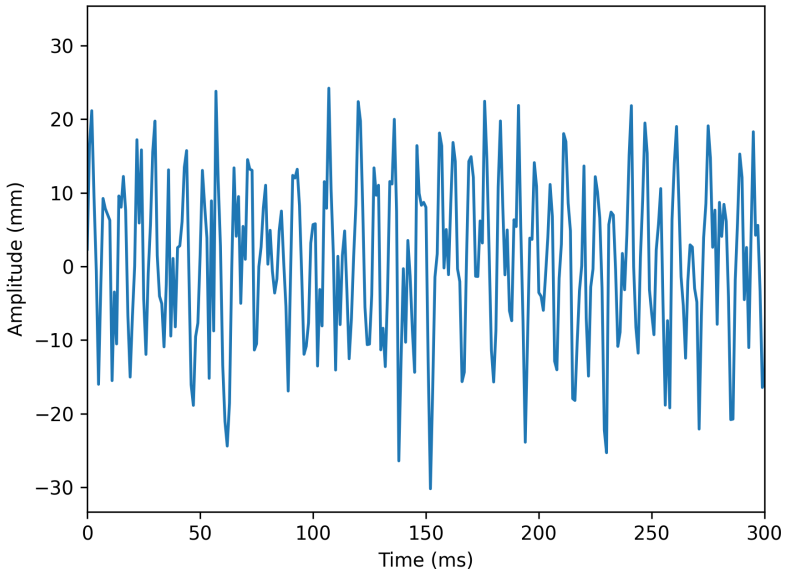

Figure 4: Unknown signal with noise. Plotting 300 points of 2000

3 Unknown Signal

Similar to the example above, we process an unknown noisy signal (4) and reconstruct the original signal. Taking the Fourier transform and creating a filter function with three Gaussian curves centered at index $117, 154, 286$, notice that the filtered signal removes any noise and isolates the dominant frequency (5T,M,B).

The peaks at these index correspond to time-periods of,

$$t_1 = \frac{2000}{117} \approx 17.0 \pm 0.1\text{ms}, \quad t_2 = \frac{2000}{154} \approx 13.0 \pm 0.2\text{ms}, \quad t_3 = \frac{2000}{286} \approx 7.0 \pm 0.5\text{ms}$$

and relative amplitudes (with respect to largest peak) of,

$$r_1 \approx 0.24 \pm 0.02, \quad r_2 \approx 0.54 \pm 0.03, \quad r_3 = 1$$

Where the uncertainties are found by the method used before (r_1 has no uncertainty since we set it as unity). However, we can only be confident in these relative amplitudes up to 85% since in Section 2 we saw that the calculated relative amplitudes is roughly 0.71 while the true relative amplitude was 0.60.

Taking the inverse Fourier transform and plotting the filtered signal and the reconstructed signal along with the original noisy signal (6). Notice that the reconstructed signal is similar to the filtered signal, however there is slight variation which is expected.

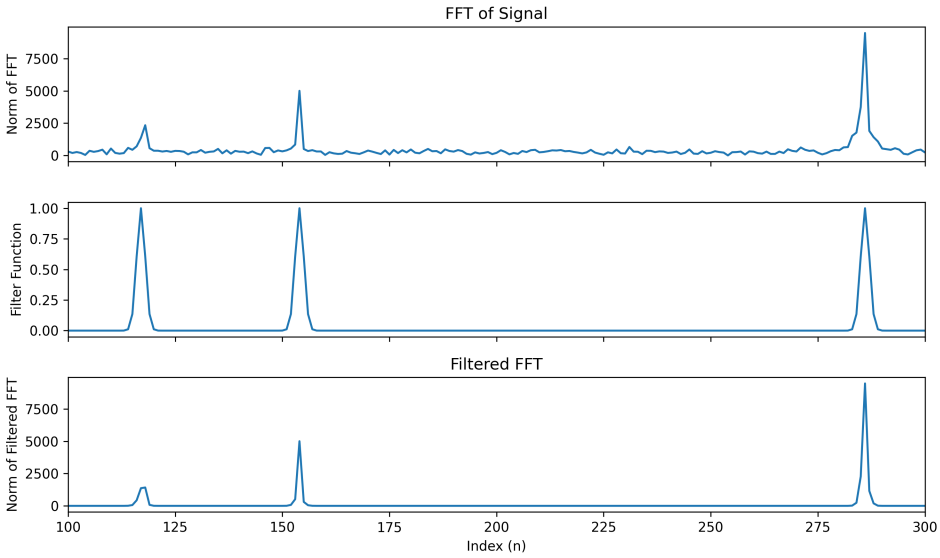

Figure 5: Top: Fourier transform of the original noisy signal along with (Middle) the filter function used to remove noise and (Bottom) the filtered signal with no noise.

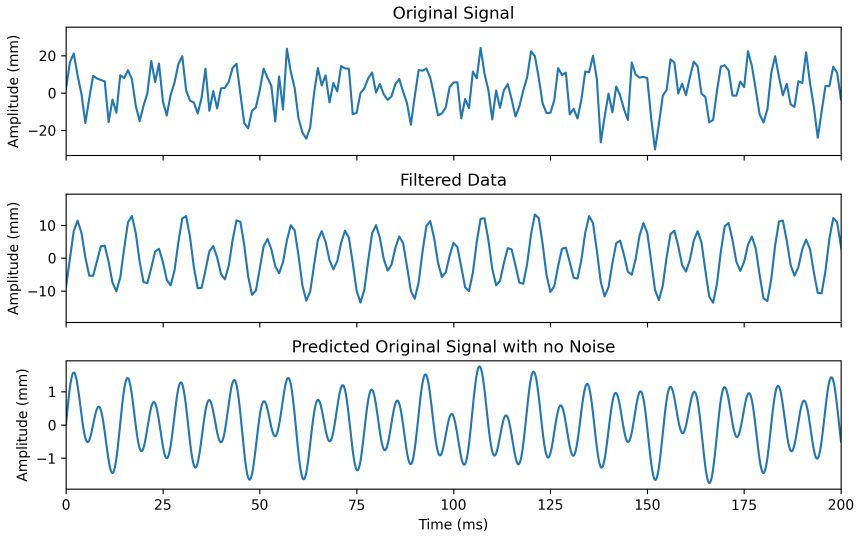

Figure 6: Top: First 300ms of the original noisy signal along with the (Middle) filtered signal and (Bottom) the reconstructed signal using frequencies and amplitudes found above.

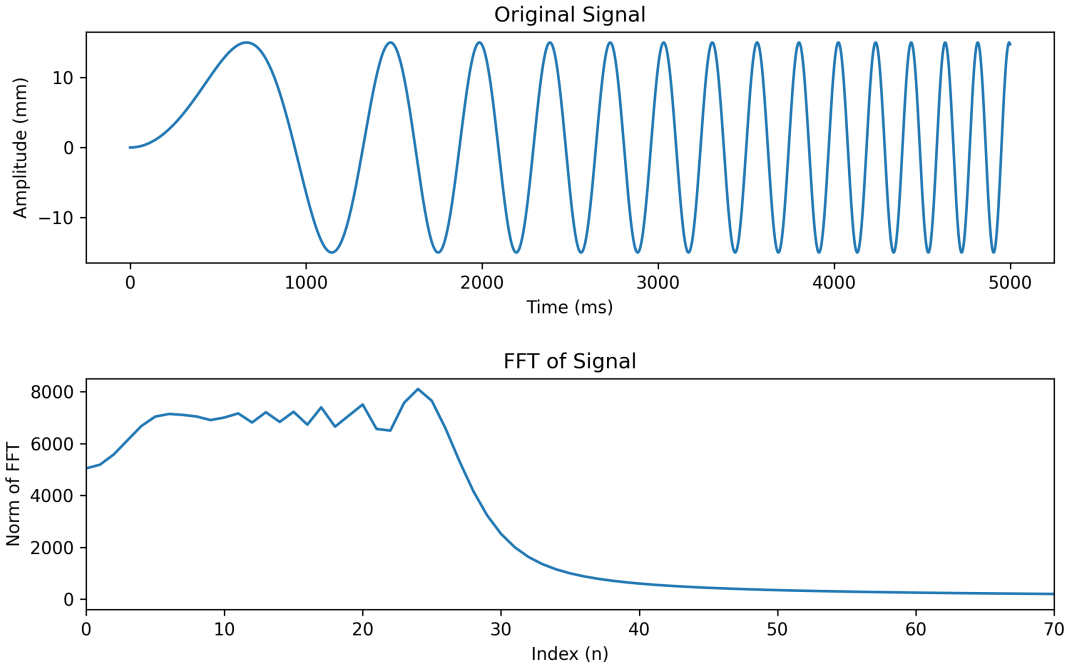

Figure 7: (Top): A signal with a linearly varying frequency and its (Bottom) Fourier transform. *Plotting a small section of the FFT since the rest of the plot contains no extra information.*

4 An Interesting Case

Although Fourier transform is used for fixed frequencies, we can take a Fourier transform of frequencies changing in time. Consider a signal where the frequency is changing linearly in time. The Fourier transform looks different to the examples we have seen before. (7). Notice that the FFT appears to have a broad range of frequencies from 0 to 25 which is expected since the signal frequency is varying in time. We also notice the oscillations in the FFT, this is the same phenomena that happens when we use a finite Fourier series to approximate a discontinuous function. In the limit as $t \to \infty$ we expect the FFT to appear as a flat horizontal line since the signal oscillates from 0 to infinite frequency.

5 Conclusion

As we can see, Fourier transform and FFT is a useful tool to analysis signals that have noise. We use FFT to generate the a plot from which we can find the relative amplitudes up to uncertainty. Similarly, from the index of the dominant peaks we can calculate the time-periods of the original signal. Using a Gaussian filter function centred at the index of dominant peaks, we can remove majority of the noise and recreate the original signal using the calculated amplitudes and frequencies.