TRACKING SUBMARINES USING THE FAST FOURIER TRANSFORM

CHENAB

Applied and Computational Mathematics, University of Washington, Seattle, WA
chenab@uw.edu

Abstract. The following paper discusses how the the problem of location estimation of a moving submarine in the Puget Sound was solved. The data available was the broad spectrum recording of acoustic pressure data. The data was analyzed using the Fourier Transform, by estimating the dominant frequency generated by the submarine. Then further de-noising and filtering was done in the Fourier domain through the use of a Gaussian Filter to extract more accurate coordinates in 3D space from the estimate the path taken by the submarine.

1. Introduction and Overview

The main problem was locating and determining the path of a submarine that was moving in the Puget sound. The only data available was the broad spectrum recording of acoustics pressure data obtained over 24 hours in half-hour increments.

The most important aspect that made this problem solvable was that the submarine emits an unknown acoustic frequency. This fact is useful because it enabled us to use the Fast Fourier Transform to analyze the frequency spectrum from the data. And upon averaging the frequency data over all time points, all other frequencies apart from the dominant frequency being emited by the submarine cancelled out by virtue of being normally distributed random noise. Then some further filtering was performed to further deemphasize frequencies apart from the submarines's dominant frequency.

Finally, after denoising and filtering, we use the inverse Fast Fourier Transform for each time point to get the point in space where the strongest signal was coming from, and connecting each discrete point over the 49 time points gives us a precise estimate for the submarines path. The following sections will go over the details of the concepts and implementation.

2. Theoretical Background

The **Fourier Transform** essentially converts any given function into a sum of sines and cosines. The formal equation for the transform (for uni-dimensional data) is the following:

$$F(k) = \frac{1}{\sqrt{2\pi}} \int_{-\infty}^{\infty} e^{-ikx}\, f(x)\, dx.$$

And the equation for the inverse transform to go from the fourier domain back to the physical domain is the following:

$$f(x) = \frac{1}{\sqrt{2\pi}} \int_{-\infty}^{\infty} e^{ikx}\, F(k)\, dk.$$

Here e^{ik} represents the sines and cosines and is derived from the Eulers formula:

$$e^{ix} = \cos(x) + i\sin(x)$$

The Fourier Transform is especially useful for time-frequency analysis because it provides a mathematical framework for transforming a problem from the physical domain to its frequency components. This, in our problem, is done through the use of the **Fast Fourier Transform (FFT)** algorithm.

We use the FFT algorithm here because it finds the Fourier transform on an interval $x \in [-L, L]$. Moreover, it is one of the most efficient algorithms, with time complexity of $O(NlogN)$, as it discretizes the

Date: February 2, 2025.

problem into smaller and smaller chunks (2^n to be exact), and hence is applicable to our problem since we have been given 64 points in each dimension.

So in effect, we use the FFT algorithm to transform the spatial pressure data into the frequency domain consisting of the frequency bins K_x, K_y and K_z, corresponding to the spatial directions of x, y and z.

Once we compute the FFT for each time point, we then average all the FFTs at each time point. We do this primarily to find out a consistent and unique frequency signature for the submarine. The primary assumption, which turns out to be quite accurate, is that all the other frequencies and white noise apart from the submarine are essentially **random and normally distributed**. This means averaging them will result in them cancelling out and the average for all of them will concentrate around the mean, which is zero. Hence all the other useless frequency signatures will be die out and the consistent and unique submarine signature will stand out, and we can easily get the K_x, K_y and K_z wavenumbers of its distinct dominant frequency signature.

While averaging enables us to determine the dominant frequency, there is a lot of noise that makes it difficult to analyze it. In these scenarios, additional filtering is often needed. A filter basically removes white noise from unwanted frequencies by reducing their strength and isolates the dominant frequency we are searching for. A **Gaussian Filter**, centered around the central frequency, is a solid example of a good filtering mechanism. Through its characteristic bell shape, all the frequencies away from the dominant frequency are attenuated, and the central (or dominant frequency) is better focused and isolated.

$$G(k_x, k_y, k_z) = e^{-\tau\left((k_x - k_{x,\text{center}})^2 + (k_y - k_{y,\text{center}})^2 + (k_z - k_{z,\text{center}})^2\right)}$$

3. Algorithm Implementation and Development

The following algorithm was implemented to extract the submarines location information from raw acoustic pressure data:

1.) Initial Setup We first load the acoustic pressure data taken over 49 time intervals of half-hour each. We define our spatial domain from $[-L, L]$ with 64 data points in each 3D direction using the **linspace** command from the **numpy** library in python. We then connect the 64 data points arrays in each direction using the **meshgrid** command.

Then we calculate the corresponding frequency grid for each coordinate using principles from Fourier Analysis. In particular, because the FFT assumes we are on a 2π periodic domain from $[0, 2\pi]$, we scale our data through the operation $\frac{2\pi}{2L}$.

2.) Data Processing Once our initial setup has been completed, we notice that our $64^3 \times 49$ dataset is flattened and contains acoustic pressure recordings from a $64 \times 64 \times 64$ grid over 49 time points. This means, for each time point, we use **reshape** command from **numpy** to convert our our data from 64^3 back into a $64 \times 64 \times 64$ 3D array. Our final signal array is thus four dimensional with shape $49 \times 64 \times 64 \times 64$.

3.) Averaging of the Fast Fourier Transform

Now, for each time frame, we compute the 3-Dimensional Fast Fourier Transform, of the acoustic pressure data in the spatial domain using the **fftn** command from the **fft** package (*fftpack*) from the **scipy** library. We also use **fftshift** to more intuitively rearrange the output of the FFT such that the zero frequency is in the center instead of the beginning, as in the case of FFT's default output.

Once we have stored the FFT output for acoustic pressure readings at each time point, we **average** the Fourier transform. The main reasoning behind it is that all the unnecessary white noise is essentially randomly and normally distributed, and averaging will essentially make the noise cancel out and only the consistent **dominant frequency signature** of the submarine will remain.

We extract the K_x, K_y and K_z by finding the **argmax** of the absolute value of our FFT average (because we also have complex arguments), and then use the **unravel-index** command from the **numpy** library to find the 3D coordinates of the maximum since **argmax** only works for the flattened array.

4.) Filtering to further de-noise the data

Once we have computed our **dominant frequency** for our submarine, we would like to further de-noise our data in the frequency domain by attenuating unwanted frequencies. We implement this using a **Gaussian Filter**, and multiplying each entry of our signal array with the filter.

5.) Inverse Fourier Transform

After filtering around our dominant frequency, we invert the FFT array back into the spatial domain using the **ifftshift** followed by **ifftn**

5.) Extracting submarine location from the filtered spatial data

After we invert the FFT, we get the "cleaned-up" spatial data, that mostly contains the submarines signal with considerably reduced noise.

Now we define a new array of shape 49×3 that stores the 3D spatial coordinates of the submarine. We get these spatial coordinates by finding the point in space for each of the 49 time frames where the signal has the **maximum magnitude**. Primarily done , as earlier, using the **argmax** and **unravel-index** commands.

5.) Plotting the submarine's path

Now that we have the submarines estimated coordinates at each time frame, we can visualize the submarines path, by using Python's **Matplotlib** library.

For the submarines path in the 3D space , we use the **scatter** function, and connect each data point with a line that shows the submarine's estimated path. The same is done to see the submarine's path in the xy plane (so that an airplane can track it)

4. Computational Results

Upon averaging the FFT at each time point, we get the following central frequency wavenumbers (rounded to three decimal places).

K_x	K_y	K_z
2.199	5.341	-6.912

Table 1. Wavenumbers of the dominant frequency in the Fourier domain

These frequencies correspond to the most prominent patterns in the Fourier domain. This means they **must** belong to the submarine because all the other random white noise tends to cancel out upon averaging. These frequencies are the basis for our Gaussian filter in subsequent signal processing because the filter we design will attenuate all the frequencies away from this central frequency. We can verify these values by seeing the strengths of signal at cross sections of the planes of central frequencies. In the Figure 1 below, we see the normalized signal strengths, each sliced at the maximum K_x, K_y and K_z. For each cross-section, it is clear that the maximum signal strength occurs at the corresponding center frequency values for K_x, K_y and K_z.

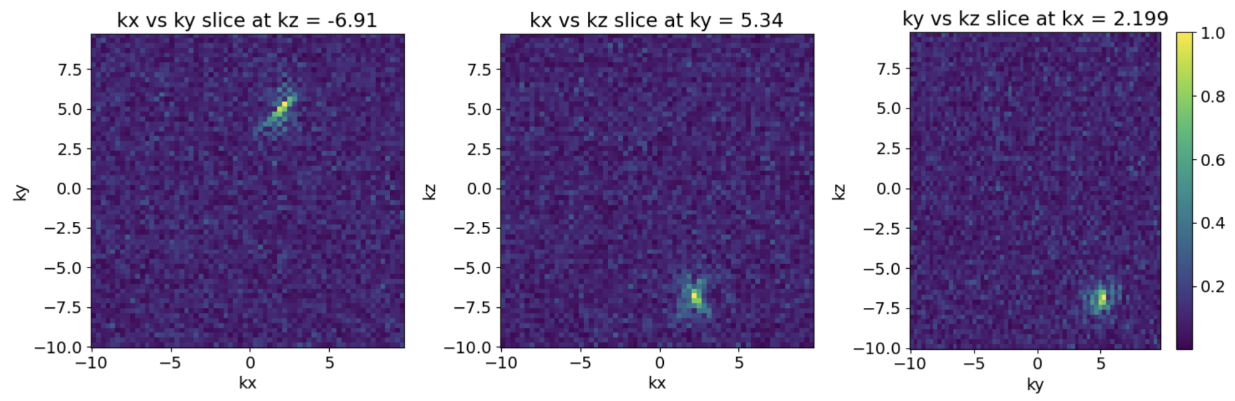

Figure 1. Normalized Cross-Sectional Views of the Averaged Fast Fourier Transform in Fourier domain. The location of the dominant frequency is visible as peaks in strength in all the three graphs

Now that we have verified our submarine's dominant frequency signature, we created a filter that attenuates values far from the particular signature, making the data cleaner. After filtering, our inverse Fourier Transform will give us a much cleaner spatial data with most of the noise removed. This enables us to plot the submarine path by finding the spatial coordinate at each time frame where the strongest signal is found and connecting them through lines. Figure 2 shows what the submarine's path in 3-Dimensional space looks like.

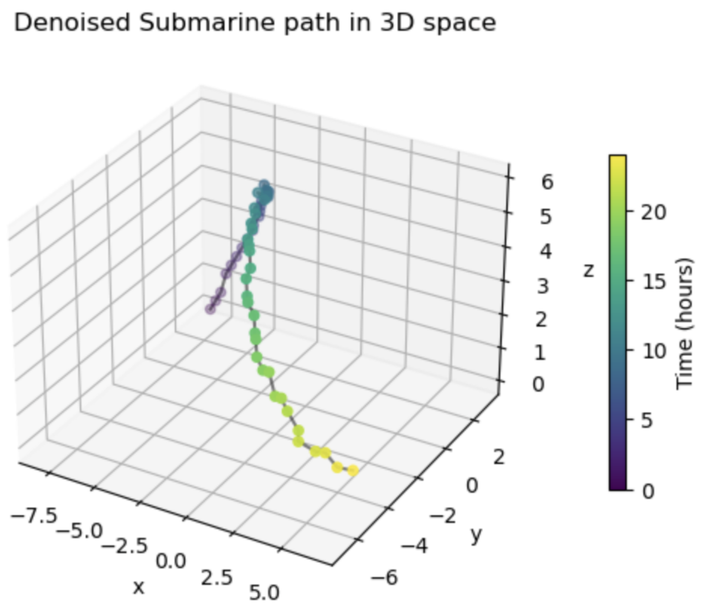

FIGURE 2. Denoised Submarine Path in 3 Dimensional Space

We can verify the validity and effectiveness of our denoising through the **Gaussian filter** by visualizing the submarine's path had no filtering been done around the dominant frequency (Figure 3). The path seems way less smoother and more jagged, showing the effects of random noise.

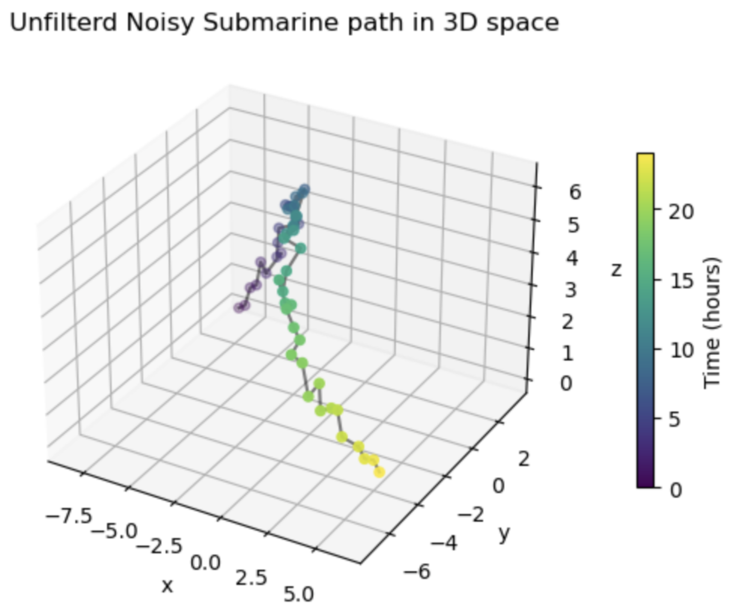

FIGURE 3. Noisy Submarine Path in 3 Dimensional Space

Finally, we can plot the denoised submarine path in the X-Y plane that a sub-tracking aircraft can be deployed. The first (x,y) value is : (-8.125, 3.125) and the final (x,y) value is (6.5625, -5.0), showing the submarines starting and ending location during the time frame.

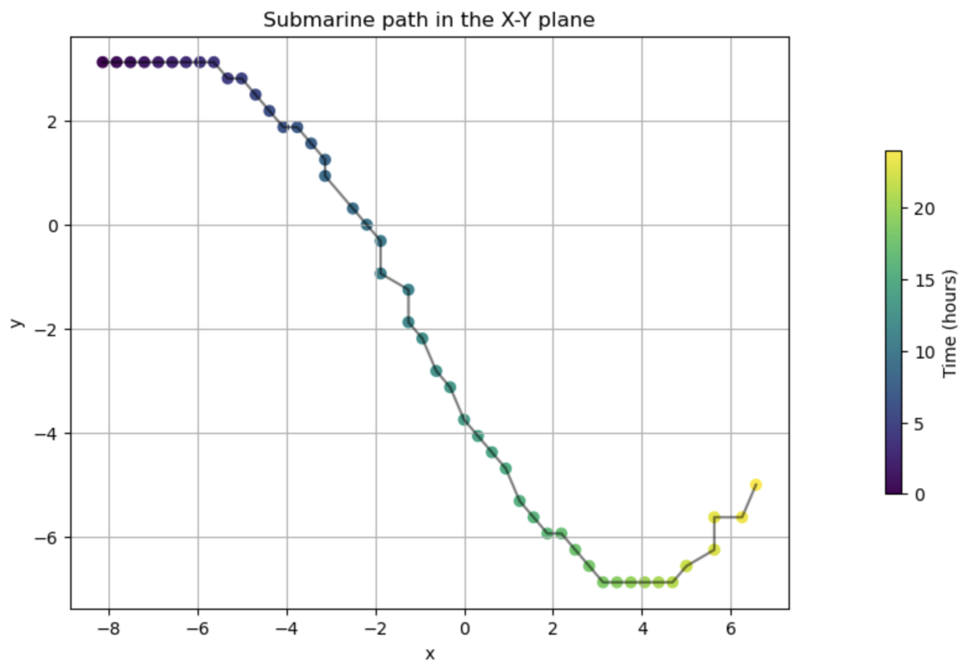

FIGURE 4. Submarine Coordinates in the XY plane

5. Summary and Conclusions

In conclusion, we effectively used the Fast Fourier Transfrom to find the submarines location in the Puget Sound from the acoustic pressure data. We had to go through the main challenge of handling the vast amount of noise in the pressure data and we utilized principles from statistics and spectral analysis, from averaging the Fourier Transform to filtering along the Gaussian. In the end, we successfully filtered out most of the noise and extracted the location of the submarine from its distinct frequency signature.

Acknowledgements

The author is thankful to Rohin Gilman for useful discussions about the conceptual understanding behind the Fast Fourier Transform. Rohin was helpful in clearing key conceptual doubts I had on the FFT. The author also is thankful to the peer Rohan Pandey for discussion and implementation of visualization techniques in Python, particularly for Task 1. .

References

1.) Kutz, J. Nathan. "Basics of Fourier Series and the Fourier Transform," "FFT Application: Radar Detection and Filtering," and "FFT Application: Radar Detection and Averaging." In Data-Driven Modeling Scientific Computation: Methods for Integrating Dynamics of Complex Systems and Big Data, 319-333. Oxford: Oxford University Press, 2013.