AMATH 582: HOME WORK 1

SUKHJIT KAUR

Department of Mathematics, University of Washington, Seattle, WA
sukhjitk@uw.edu

Abstract. The path of a submarine moving in the Puget Sound is predicted from obtained data with unknown acoustic frequency. Using Fourier analysis and implementation in Python, the center frequency was found and used to create a filter. By applying the filter to the original noisy data, the submarine's trajectory was calculated and the submarine could be located. The original noisy data was compared to the filtered data, revealing a much sharper signal.

1. Introduction and Overview

A submarine is known to be moving in the Puget Sound, emitting an unknown acoustic frequency that we need to detect to locate the submarine. Acoustics pressure data was measured over a 24-hour period in half-hour intervals. However, this 3-dimensional data, acquired over a 64x64x64 grid is noisy and requires filtering to be more useful. By using Fourier analysis, and filtering methods, namely Fast Fourier Transform and Gaussian filtering using Python, the central frequency can be determined and used to reduce the noise in the acoustics measurements. The filtered signal can be used to track the path of the submarine and locate its whereabouts. The results can be applied to establish a sub-tracking solution that can track the submarine in the future.

2. Theoretical Background

The Fourier Transform (FT) is an integral transform that can be used to realize the spatial frequency of noisy measured signal data.
Fast Fourier Transform (FFT) is a more efficient method of computing the forward and backward transforms, and was developed by Cooley and Tukey [1]. FFT has a computational complexity of $O(NlogN)$, due to its efficient method of finding coefficients using symmetry, compared to previously known methods that clocked in at $O(N^3)$. Specifically, this is achieved by discretizing the interval $[-L, L]$ into 2^n points. A 2-D, 3-D, or N-D FT can be used for higher-dimensional Fourier transforms, as is done for this dataset.

The 3-D FT is calculated by the following [3]:

$$\hat{f}(k_x, k_y, k_z) = \frac{1}{(\sqrt{2\pi})^3} \int_{-\infty}^{\infty} \int_{-\infty}^{\infty} \int_{-\infty}^{\infty} f(x, y, z) e^{-i(k_x x + k_y y + k_z z)} dx dy dz \tag{1}$$

Formally, we see the transform happens over the entire real line, but for the computational domain used in Python, we use a finite domain ranging from some -L to L.

Date: January 21, 2025.

Furthermore, a 3-D Gaussian filter can be used to de-noise the given dataset. This can be accomplished using the product of Gaussian filters along each dimension, k_x, k_y, k_z, calculated as below, where σ represents the variance. [3, 2]:

$$G(k_x, k_y, k_z) = e^{-\tau((k_x-k_x^*)^2+(k_y-k_y^*)^2+(k_z-k_z^*)^2)}, \tau = \frac{1}{2\sigma^2} \tag{2}$$

3. Algorithm Implementation and Development

To computationally find the Fast Fourier Transform of the noisy acoustic pressure data, various packages and libraries were implemented within Python - most importantly, Numpy. Matplotlib was used for visualizing the data and analysis.
Using numpy.fftn, the 3-dimensional FFT is calculated over the given time period for the spatial data of the submarine. A filter is then applied to the data in the frequency domain, and numpy.ifftn is used to reveal the submarine path. In the Python implementation of FFT, the k-values start at zero with alternating signs, so the k-mesh must be shifted using numpy.fftn.fftshit to center the values of k at zero. The FFT is also manually scaled by a factor of $1/N$, since numpy natively scales the inverse FFT. The filter is used to denoise the measurements, and is created using the Gaussian equation in 3 dimensions, and the center-frequency of the data.

4. Computational Results

Using the averaged Fast Fourier Transform of the noisy pressure data, the center-frequency was found at $(7.8540, 3.7699, 9.4248)$, (Table 1). This is the dominant frequency, or signature, of the submarine highlighted in Figure 1. Using this frequency, the Gaussian 2 filter was created using $\sigma = 5$ and implemented to produce the de-noised data-set. The effect of the filter on the trajectory of the submarine over the 24-hour period is shown in Figure 2. Both the noisy and filtered trajectories are represented in 2- and 3-D.

k	index	center-frequency
k_x^*	57	7.8540
k_y^*	44	3.7699
k_z^*	62	9.4248

Table 1. Center-frequency of the submarine

5. Summary and Conclusions

Through averaging of the Fast Fourier Transform of the dataset, the center-frequency was successfully determined. The subsequent creation of the Gaussian filter allowed for decent filtering of the data and resulted in a well-represented submarine trajectory. Though the effects of the Gaussian filter on the acoustic pressure data are noticeable, there is possibility for greater clarity of the submarine trajectory 2 using alternate filtering methods with other transforms such as Gabor or Wavelet transforms. This is especially true for more complex datasets, or where there may be more than one dominant frequency. Filtering using one dominant frequency may result in inaccurate filtering over the entire dataset. For such a case, a combination of filters could be implemented and evaluated at each time step.

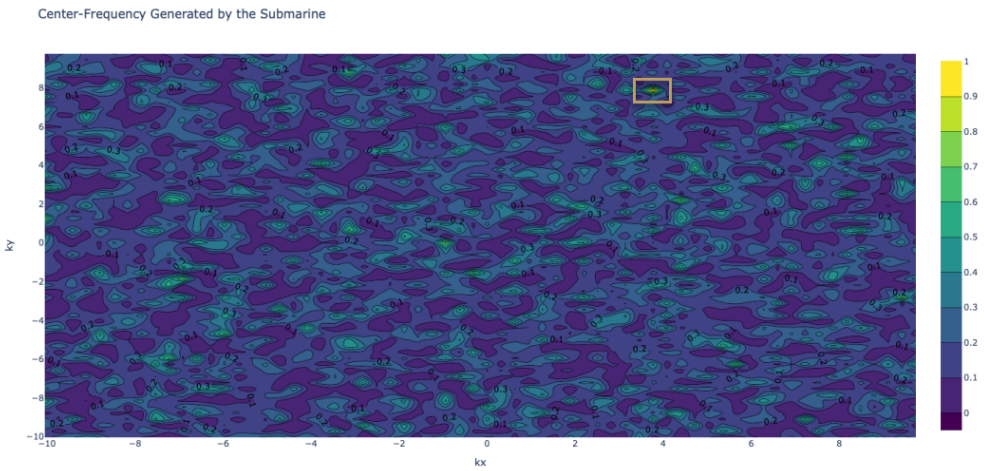

FIGURE 1. The center-frequency of the submarine at (k_x^*, k_y^*, k_z^*) represented by a contour map at $z = k_z^*$

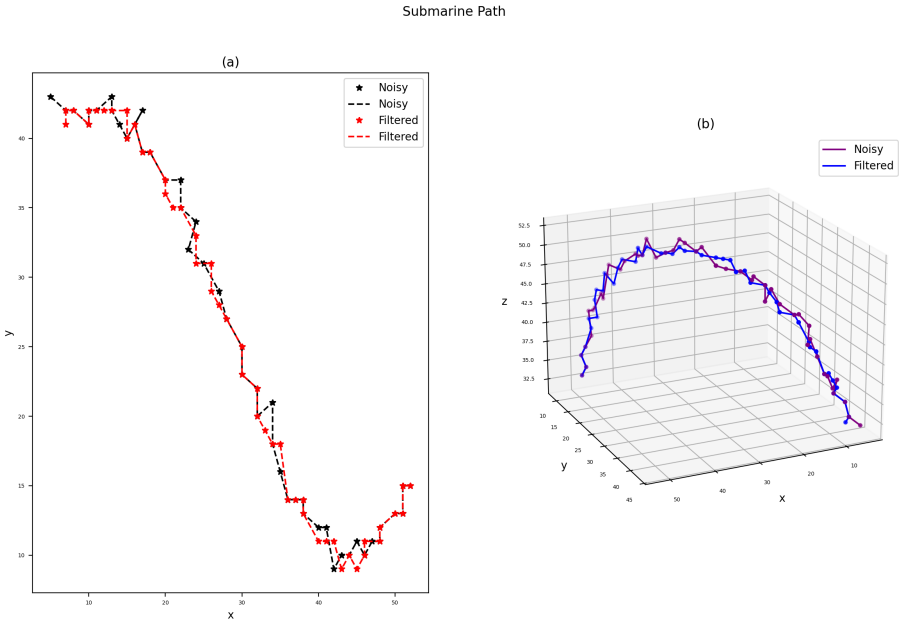

FIGURE 2. Submarine trajectory plotted over 24-hours, before and after filtering. (a) shows the path in 2-D, where (b) gives a 3-D representation.

The results were limited by the number of data-points provided in the data-set. Future work could entail applying the filter to a data-set with more data-points (measurements taken more frequently, or over a longer period of time) and creating filters using other transforms.

ACKNOWLEDGEMENTS

Special thanks to Saba Heravi for providing guidance on setting up the problem and to Eli Shlizerman for clarified explanations of Fourier Series and Fourier Transform. The author

is also appreciative of Large Language Models and it's assistance in explaining code usage that allowed for proper plotting and debugging.

References

[1] J. W. Cooley and J. W. Tukey. An algorithm for the machine calculation of complex fourier series. *Mathematics of Computation*, 19.90(1):297–301, 1965.

[2] S. Heravi. Amath 582 methods for data analysis - homework 1 - ta slides. Supplementary notes from TA for AMATH 582.

[3] J. Kutz. *Methods for Integrating Dynamics of Complex Systems and Big Data*. Oxford, 2013.