ACOUSTIC PRESSURE DENOISING

RYAN MOUW

Applied Mathematics Department, University of Washington, Seattle, WA
ryan.jay.mouw@gmail.com/rmouw@uw.edu

Abstract. We are presented with the acoustic pressure of a cubic section of the ocean in which a submarine is located. The goal is to find the location of the submarine at 49 points in time, amongst the noisy acoustic data, in order to plot the submarine's course. Fast Fourier Transform (FFT) is used to isolate the central frequency of the acoustic signal emitted from the submarine. The central frequency is then used to filter the noisy data via a Gaussian filter. This method produced a clear trajectory of the submarine over the given time interval, with little noise interference.

1 Introduction and Overview

To begin, we are presented with a 262144 x 49 matrix of acoustic data. The rows in this matrix represent the points in a 64^3 cubic unit of ocean. The columns of the matrix represent 49 different moments in time, which are understood to be half hour increments over 24 hours. The values of each point in the matrix represent the acoustic pressure at that corresponding time and location.

The problem we face is that the ocean is naturally noisy, so the noise emitted by the submarine will be somewhat masked. In this problem, we are assuming that the additional noise in the data is Gaussian with mean zero. To solve this problem we use the known fact that adding Gaussian random noise with mean zero to data is equivalent to adding random noise to the FFT of the data.[3] This will be discussed further in the proceeding section.[2]

2 Theoretical Background

The main tool used to solve this problem is the Fast Fourier Transform (FFT). To understand the FFT, we first discuss the Fourier Series and Fourier Transform. The Fourier Transform is the mathematical transform that transitions functions that depend on space and time, into functions that depend on frequency. [1] The Fourier Series is a means of representing any arbitrary function as a sum of weighted sine and cosine waves. Larger weights (sine and cosine coefficients) tell us that the corresponding frequencies are heavily depended upon when recreating the arbitrary function. The FFT, then, is an algorithm that performs the Fourier Transform very quickly and in turn gives us very fast access to the heavily weighted, important coefficients. The FFT computes the coefficient for the corresponding k^{th} dimensional frequency for a function $f : \mathbb{R}^k \to \mathbb{R}$;

$$f(n) = f(x_n) \qquad \text{for} \quad n = 0, 1, ..., N-1$$

$$C_k = \sum_{n=0}^{N-1} f(n) \exp\left(-2\pi i \frac{nk}{N}\right) \qquad [4]$$

Date: January 28, 2022.

The Inverse Fast Fourier Transform (IFFT), as the name suggests, takes a function that depends on frequency, and maps it into the domain of space or time. This is useful because it allows the filtering of values in the Fourier domain, which then produces smoother and cleaner waves in time and space after the IFFT is applied.

The next crucial piece is understanding that adding random noise (which has a Gaussian distribution and mean zero) to a signal is equivalent to adding the same distribution of random noise to the function's FFT.[1] This fact allows to us filter out some of the noise in the FFT by averaging the coefficients returned from FFT over multiple time steps. [3] In a sense, the noise cancels itself out, leaving a spike in the Fourier domain at the location of any frequencies that are not Gaussian randomly distributed in the space.

3 Algorithm Implementation and Development

To find the central frequency I first reshaped the raw acoustic data matrix from 262144 x 49 to 64 x 64 x 64 x 49. I then took the FFT of each of the 49, 64^3 cubes of data. To average out the Gaussian noise, I averaged each of the 49 cubes element wise. The resulting 3D cube is the FFT averaged over each time step (0 through 48). The indices associated with the largest value in the averaged FFT represent the 3D central frequency, or the 3D frequencies that the submarine is emitting.

Once the central frequency is obtained, a Gaussian filter centered at the same coordinates as the central frequency within the Fourier domain is created. The FFT of each time step (0 through 48) is then multiplied by the values of the Gaussian filter, element wise. This multiplication dampens all of the values not immediately surrounding the central frequency in the Fourier domain. Through trial and error, it was determined that a Gaussian filter with $\sigma = 3$ was sufficient to smooth out the noise.

Once the filter is applied to the FFT of each time step, the Inverse Fast Fourier Transform (IFFT) was then applied to each FFT. As the name suggests, the IFFT transforms the data from the frequency domain, back into (in our case) the spatial domain. The result is a 64^3 cube of acoustic data with the random noise greatly reduced. The location in each cube with the largest acoustic pressure is then interpreted as the location of the submarine. By taking the index with the maximum value across each time step, the submarine's path is then revealed.

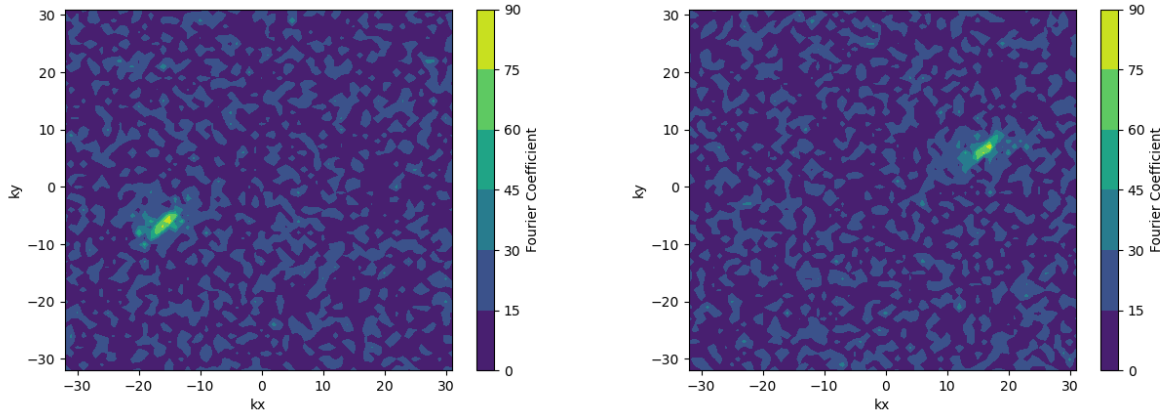
The matrix manipulation and fft/ifft algorithms are done in code using using the NumPy and SciPy python packages.[6][2]

4 Computational Results

4.1 Average FFT and the Central Frequency

To help locate the central frequency of the submarine we use the fact that adding random noise that has a Gaussian distribution with mean zero to a function is equivalent to adding the Gaussian distributed random noise with mean zero to the FFT of a function.[3] In our case, we summed and averaged the FFT of each of the 49 time steps. With enough time steps to average over, the effect is that of canceling out much of the noise in the Fourier domain. In this problem, the averaged Fourier domain revealed two clear peaks. The Fourier domain's axes are shifted within the 3D Fourier space, but is centered at $(0, 0, 0)$. Using this center, the peak values in the averaged FFT were at indices $(17, 7, -22)$ and $(-15, -6, 23)$. The two peaks are a result of the data being real (as opposed to complex or imaginary), which means the Fourier Transform is symmetric. These two peaks are reflections through the origin, and correspond to the (x, y, z) frequencies that the submarine is emitting.

FIGURE 1. Peak values in Averaged FFT



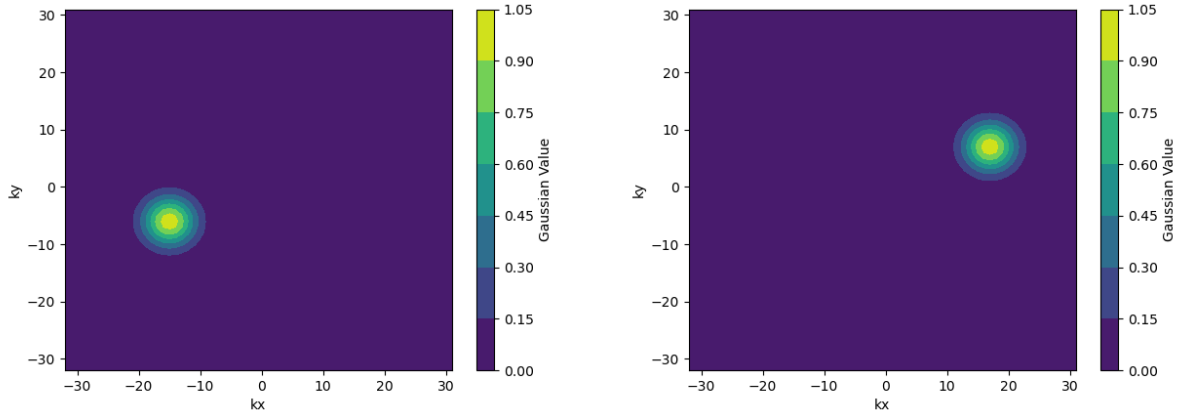
(A) $kz = 23$ (B) $kz = -22$

(C) All visuals used in this document were created using the MatPlotlib python package.[5]

Now that we have found the two central frequencies (the (x, y, z) components of the peaks in the Fourier domain) we can design a filter that will eliminate much of the noise surrounding the frequencies. We do this by defining a 3D Gaussian curve in the same coordinate space as the Fourier domain. For this case, we will want two Gaussian curves, one centered at $(17, 7, -22)$ and another centered at $(-15, -6, 23)$. The FFT of each time step can now be multiplied by this filter, element wise, to eliminate all the noise that is far from the central frequencies. The effect of this multiplication is that all of the values in the Fourier domain are multiplied by a value between 0 and 1, with the central frequency being multiplied by 1, and indices farther away from the central frequency being multiplied by zero. Two different 2D z-slices of the Gaussian filter is shown in Figure [3].

Through trial and error, a value of $\sigma = 3$ was found to be sufficient in eliminating noise in the Fourier domain, and lead to a well established trajectory of the submarine.

FIGURE 3. Gaussian filter $\sigma = 3$

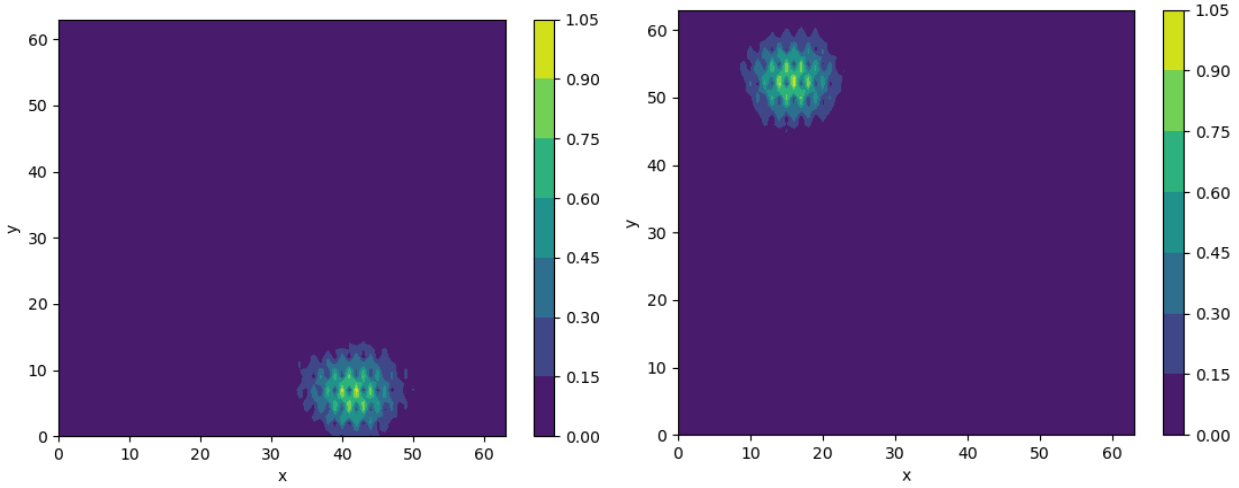


(A) $kz = 23$ (B) $kz = -22$

4.2 The Reconstructed Signal

Once the FFT of each time step has been multiplied by the Gaussian filter, the IFFT is then used to transform the data back into the spatial domain, but with much less noise. Once each cube is transformed back into the spatial domain, the area of largest acoustic disturbance becomes very clear. Figure [5] shows 2D slices of the reconstructed signal at different moments in time. The signal has been normalized against the maximum value. As you can see in Figure [5], the reconstructed signal clearly shows a 3-dimensional traveling wave. This is the location of the submarine. By looking at each time step's reconstructed signal, we are able to piece together the submarine's path over the 24 hour period.

FIGURE 5. Reconstructed Filtered Acoustic Pressure Data



(A) $t = 0,\ z = 32$ (B) $t = 48,\ z = 35$

4.3 Submarine Path

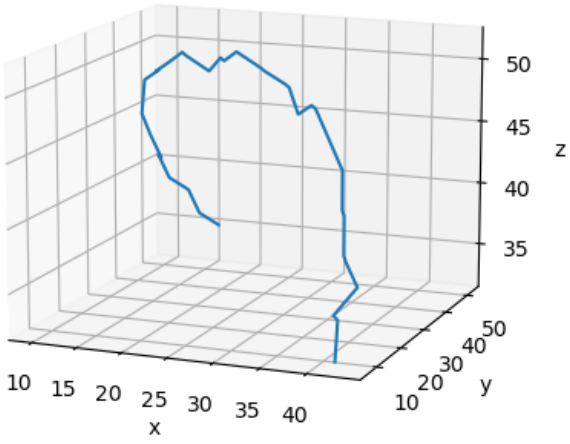
The (x, y, z) coordinates of the submarine's path are given for each time step in the Table [1]. Figure [7] plots the submarines path in 2D, representing the surface of the ocean. This data can be used to deploy submarine tracking aircraft in the future. Accompanying the 2D plot, is a 3D plot of the submarines path [7]. The 3D plot reveals the large amount of variation in the z-axis of the submarine, and also helps put the 2D plot in context.

The 3D plot in Figure [7] shows that the submarine path has not been interpolated smoothly between the different points in time. This can be explained by a number of factors. First, the acoustic pressure is taken in 30 minute intervals. The submarine has the potential to travel a fair distance in 30 minutes, and this alone would lead to a jagged path, as the plotting mechanism assumes a direct path from point to point. Finally, we cannot assume that the submarine is traveling in a smooth trajectory. The submarine could in fact be traveling using sharp, distinct turns to avoid detection. Both of these variables combined give some credence to the jagged nature of both the 2D and 3D plots.

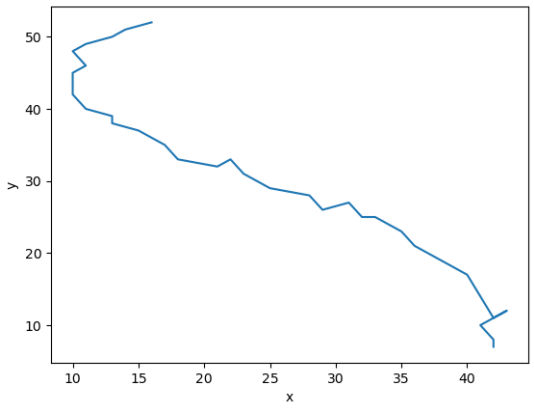
Time step	0	1	2	3	4	5	6
(x,y,z)	(42,7,32)	(42,7,32)	(42,8,32)	(41,10,32)	(42,11,32)	(43,12,32)	(42,11,32)
Time step	7	8	9	10	11	12	13
(x,y,z)	(39,14,32)	(42,14,32)	(42,14,32)	(42,17,32)	(45,17,32)	(45,17,32)	(47,19,32)
Time step	14	15	16	17	18	19	20
(x,y,z)	(38,19,47)	(36,21,49)	(36,21,49)	(35,23,49)	(33,25,48)	(32,25,50)	(31,27,50)
Time step	21	22	23	24	25	26	27
(x,y,z)	(29,26,51)	(28,28,51)	(25,29,52)	(25,29,52)	(23,31,51)	(22,33,51)	(21,32,50)
Time step	28	29	30	31	32	33	34
(x,y,z)	(18,33,51)	(18,33,51)	(17,35,51)	(15,37,50)	(15,37,50)	(13,38,49)	(13,39,49)
Time step	35	36	37	38	39	40	41
(x,y,z)	(11,40,48)	(11,40,48)	(10,42,45)	(10,42,45)	(10,42,45)	(10,45,43)	(10,45,43)
Time step	42	43	44	45	46	47	48
(x,y,z)	(11,46,41)	(10,48,41)	(11,49,39)	(13,50,38)	(13,50,38)	(14,51,36)	(16,52,35)

TABLE 1. The table above shows the (x,y,z) coordinates of the submarine at each of the 49 time steps (Labeled 0 through 48)

FIGURE 7. Submarine Path



(A) 3D plot of Submarine Path



(B) 2D Plot of Submarine Path

5 Summary and Conclusions

In conclusion it is seen that filtering the Fourier domain of the acoustic signals worked well in providing a denoised path of the submarine. It is important to point out, however, that the solution relied heavily on the fact that Gaussian noise can be averaged to zero. In future scenarios where the noise is not Gaussian, or perhaps the case where other frequencies also existed in the data, further techniques would be needed to isolate the central frequency. Another point to consider is the presence of complex values in the Fourier Domain. This problem did not rely very heavily on complex values to reconstruct the signal, which meant that the filtering of complex values was not considered with great care or detail. In the future, this may not be the case, and complex values in the Fourier domain could play a much larger role in signal reconstruction.

Acknowledgements

Many thanks to Roman Levin, a TA in the Applied Mathematics Department at University of Washington-Seattle, for his help in conceptual matters surrounding the interpretation of the Fourier space. I would like to thank Nalu Zou, a peer in the AMATH 482 class, with guidance on the defining of a 3-dimensional Gaussian filter. I would also like to thank Ava Mistry, another peer in AMATH 482/582, for her help with locating references to Gaussian noise in the Fourier Domain. Lastly, I would like to thank Anni Li, another peer in AMATH 482, for her help in implementing the NumPy.fftn and NumPy.ifftn packages.

References

[1] S. Brunton and N. Kutz. *Data Driven Science Engineering*. databook.uw.edu, 2017.

[2] C. R. Harris, K. J. Millman, S. J. van der Walt, R. Gommers, P. Virtanen, D. Cournapeau, E. Wieser, J. Taylor, S. Berg, N. J. Smith, R. Kern, M. Picus, S. Hoyer, M. H. van Kerkwijk, M. Brett, A. Haldane, J. F. del Río, M. Wiebe, P. Peterson, P. Gérard-Marchant, K. Sheppard, T. Reddy, W. Weckesser, H. Abbasi, C. Gohlke, and T. E. Oliphant. Array programming with NumPy. *Nature*, 585(7825):357–362, Sept. 2020.

[3] B. Hosseini. High-dimensional extensions amp; image processing. University of Washington-Seattle (LOW 216), Jan 2022. AMATH 482/582.

[4] B. Hosseini. Signal processing with dft. University of Washington (LOW 216), Jan 2022.

[5] J. D. Hunter. Matplotlib: A 2d graphics environment. *Computing in Science & Engineering*, 9(3):90–95, 2007.

[6] P. Virtanen, R. Gommers, T. E. Oliphant, M. Haberland, T. Reddy, D. Cournapeau, E. Burovski, P. Peterson, W. Weckesser, J. Bright, S. J. van der Walt, M. Brett, J. Wilson, K. J. Millman, N. Mayorov, A. R. J. Nelson, E. Jones, R. Kern, E. Larson, C. J. Carey, İ. Polat, Y. Feng, E. W. Moore, J. VanderPlas, D. Laxalde, J. Perktold, R. Cimrman, I. Henriksen, E. A. Quintero, C. R. Harris, A. M. Archibald, A. H. Ribeiro, F. Pedregosa, P. van Mulbregt, and SciPy 1.0 Contributors. SciPy 1.0: Fundamental Algorithms for Scientific Computing in Python. *Nature Methods*, 17:261–272, 2020.