AMATH 482 Assignment 1: A Submarine Problem

Matthew Mangione

January 27, 2021

Abstract

Given the acoustic frequencies of a volume representing the Puget Sound over a 24 hour period, we were tasked with locating a new class of submarine which emits a specific, unknown frequency. This required us to analyze the data-set in Fourier space, averaging the data to reveal the signature frequency of the submarine. From there, a Gaussian filter was applied to the Fourier space data, eliminating the noise of the initial data-set, allowing us to find the location of the submarine for each sample in time.

1 Introduction and Overview

A new submarine technology has been deployed to the Puget Sound. This submarine emits a particular frequency, which is initially unknown. In order to locate this submarine and its frequency, a broad range of acoustic data is collected every 30 minutes over a period of 24 hours. Each sample in time includes measurements discretized throughout a volume representing the Puget Sound. Specifically, the body of water is approximated as a 64 x 64 x 64 volume.

The data provided is naturally very noisy, which provides the difficulty in extracting the desired information. In order to locate the submarine in space, we first need to discover its signature frequency. This is accomplished by averaging the Fourier transforms of every sample in time. With the signature frequency discovered, a Gaussian filter can be applied to this location in Fourier space. This eliminates extraneous frequencies responsible for most of its noise in our data. Then, the filtered data can be converted back to the time-domain, where only the frequencies matching our signature frequency will remain. From here, we can easily locate the coordinates of the submarine in each sample by seeking the highest magnitude of the signal.

2 Theoretical Background

Our methodology is largely reliant on analyzing our data in Fourier space. A function can be represented in this domain by rewriting it in terms of sines and cosines. This is done using the Fourier series shown below.

$$f(x) = \frac{a_0}{2} + \sum_{i=1}^{\infty} (a_n \cos nx + b_n \sin nx) \quad x \in [-\pi, \pi]. \tag{1}$$

The Fourier series is useful because it allows for discontinuous functions with a finite domain. However, our discretized data is not presented as a function, and requires us to approximate the Fourier series. We will utilizing the Discrete Fourier Transform, which uses a finite number of frequencies. Because high frequencies are potentially hidden between our discretized points, there is no way to detect said frequencies, and thus the number of frequencies is capped. Given a sequence of N equally spaced points, the Discrete Fourier Transform returns a sequence of points which approximate its Fourier Series.

$$\hat{x_k} = \frac{1}{N}\sum_{n=0}^{N} x_n(\cos\frac{2\pi kn}{N} + i\sin\frac{2\pi kn}{N}) = \frac{1}{N}\sum_{n=0}^{N} x_n e^{\frac{2\pi ikn}{N}} \tag{2}$$

3 Algorithm Implementation and Development

3.1 Initialization & Discretization

First, we must reshape the inputted data from a vector into 49 samples of measurements in 3 spacial dimensions. This is accomplished using `reshape` command, resulting in a 64 x 64 x 64 x 49 array. We then discretize our spacial domain, using the `meshgrid` command to map our array coordinates to their proper location in space. Similarly, we initialize a mapping to spectral space, scaling the wavenumbers **k** by 2π to fit our spacial domain, after correcting the indexing provided by the MATLAB implementation of the Discrete Fourier Transform.

3.2 The Fast Fourier Transform

Developed in 1960 by Cooley and Tukey, the Fast Fourier Transform (FFT) is an algorithm that performs the Discrete Fourier Transform at a significantly faster rate. When the domains of the inputted sequences are discretized into 2^n points, the speed of the algorithm improves to *O(NlogN)*. Because of this, the FFT is the premiere Fourier transformation implemented in MATLAB, and the algorithm we will use to analyze our data in frequency space. Because our data spans 3 dimensions for each sample in time, we use the multidimensional FFT implemented by the `fftn` command in MATLAB. Additionally, this algorithm shifts the domain of each dimension, inverting both sides of each axis around zero. To account for this, the `fftshift` and `ifftshift` commands can be used to reorient the spectral data.

3.3 Averaging Spectral Data over Time

In order to find the signature frequency of the submarine, data must be transformed to Fourier space using `fftn`. Due to the Fourier Uncertainty Principle, there is no information about the submarine's time and position, but only a breakdown of the frequencies incorporated. Because the submarine is emitting the same signal in every sample, every realization in time will have the same frequency present, while the supposed white noise surrounding it should vary normally around zero. Then by averaging each sample of our data in frequency space, our signature frequency compounds, while every other frequency should approach zero. Once the average is found, it needs to be shifted using `fftshift` to align with our spectral domain. From there, the maximum can be taken using `[max, index] = max()`, which also provides the index of the max frequency. This is then translated to coordinates of the signature frequency by using the `ind2sub` command and inputting its results to our mapping of spectral space.

3.4 Filtering with the Gaussian

Now that the signature frequency of the submarine is known, it can be used to filter out the unimportant frequencies from each sample. This will eliminate noise such that only the locations emitting the signature frequency will remain when converted back to the time domain. First, a Gaussian is created over the spectral domain, centered around the coordinates of the signature frequency.

$$g(x, y, z) = e^{-\tau((x-x_0)^2+(y-y_0)^2+(z-z_0)^2)} \tag{3}$$

This Gaussian is evaluated at all discretizations of the spectral domain, and then multiplied by the spectral data for a given time sample to produce the filtered data. To return to the time-domain, `ifftshift` and `ifftn` are used to reverse the previous transformations to the spectral data. In the time domain, peaks in the filtered data correspond to where the signature frequency is being emitted, i.e., the submarine. Therefore the max value for each time sample of the filtered data corresponds to the location of the submarine, which can be found using the `max` command similar to the previous section.

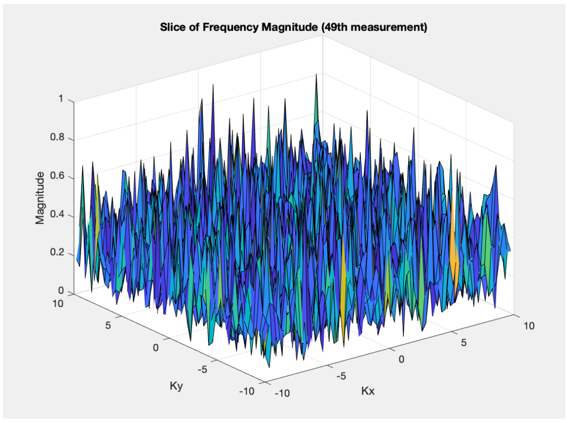

Figure 1: Subset of spectral data at t=1, showing the z-axis slice of the signature frequency.

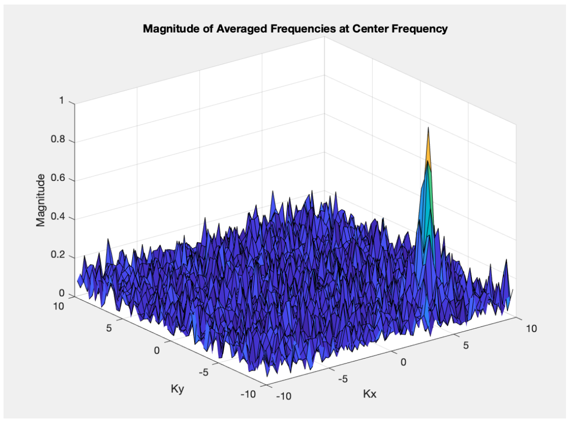

Figure 2: Subset of averaged spectral data showing the signature frequency.

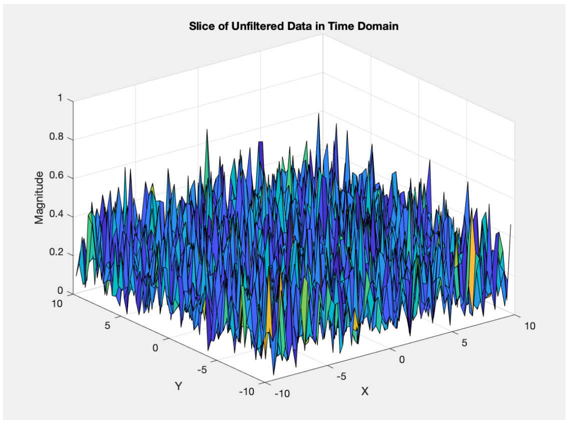

Figure 3: A Subset of the unfiltered data at t=49, showing the z-axis of the location of the submarine.

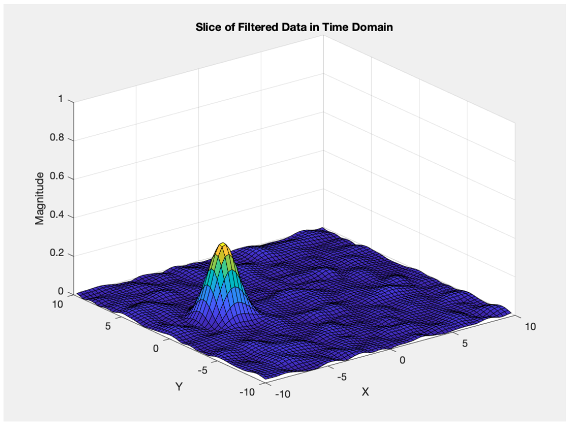

Figure 4: The same subset of Figure 3 but after applying the Gaussian filter.

4 Computational Results

As expected, averaging all the time samples in the Fourier space succeeded in compounding the desired frequency and removing the white-noise present in the data. Figure 1 shows the location of the central frequency at a single sample in time amidst the noise of the other frequencies present. Figure 2 shows this same slice of spectral domain, but after averaging all the spectral data samples together. The frequency highlighted by this peak corresponds to the location (5.3407, -6.9115, 2.1991) in the spectral domain, which represents the wavenumbers associated with the signature frequency on each axis.

With the frequency of the submarine found, the Gaussian can be created and is evaluated at every discretization in the spectral domain. After applying this filter and returning to the spacial domain, the given sample is now absent of noise. Figures 3 and 4 demonstrate the effect of applying the Gaussian filter around our signature frequency. The location of each peak for each measurement in time was determined using the `max` command, and the resulting coordinates in space were recorded. These can be viewed in Table 1. Figure 5 plots these coordinates, which when viewed together create a feasible trajectory for the submarine. The most recent location of the submarine is x = -5.0, y = 0.9375, which is our recommendation for sending the tracking aircraft.

Time	X	Y	Z
1	3.1250	0.0000	-8.1250
2	3.1250	0.3125	-7.8125
3	3.1250	0.6250	-7.5000
4	3.1250	1.2500	-7.1875
5	3.1250	1.5625	-6.8750
6	3.1250	1.8750	-6.5625
7	3.1250	2.1875	-6.2500
8	3.1250	2.5000	-5.9375
9	3.1250	2.8125	-5.6250
10	2.8125	3.1250	-5.3125
11	2.8125	3.4375	-5.0000
12	2.5000	3.7500	-4.6875
13	2.1875	4.0625	-4.3750
14	1.8750	4.3750	-4.0625
15	1.8750	4.6875	-3.7500
16	1.5625	5.0000	-3.4375
17	1.2500	5.0000	-3.1250
18	0.6250	5.3125	-2.8125
19	0.3125	5.3125	-2.5000
20	0.0000	5.6250	-2.1875
21	-0.6250	5.6250	-1.8750
22	-0.9375	5.9375	-1.8750
23	-1.2500	5.9375	-1.2500
24	-1.8750	5.9375	-1.2500
25	-2.1875	5.9375	-0.9375
26	-2.8125	5.9375	-0.6250
27	-3.1250	5.9375	-0.3125
28	-3.4375	5.9375	0.0000
29	-4.0625	5.9375	0.3125
30	-4.3750	5.9375	0.6250
31	-4.6875	5.6250	0.9375
32	-5.3125	5.6250	1.2500
33	-5.6250	5.3125	1.5625
34	-5.9375	5.3125	1.8750
35	-5.9375	5.0000	2.1875
36	-6.2500	5.0000	2.5000
37	-6.5625	4.6875	2.8125
38	-6.5625	4.3750	3.1250
39	-6.8750	4.0625	3.4375
40	-6.8750	3.7500	3.7500
41	-6.8750	3.4375	4.0625
42	-6.8750	3.4375	4.3750
43	-6.8750	2.8125	4.6875
44	-6.5625	2.5000	5.0000
45	-6.2500	2.1875	5.0000
46	-6.2500	1.8750	5.6250
47	-5.9375	1.5625	5.6250
48	-5.3125	1.2500	5.9375
49	-5.0000	0.9375	6.5625

Table 1: Coordinates of the submarine every 30 minutes.

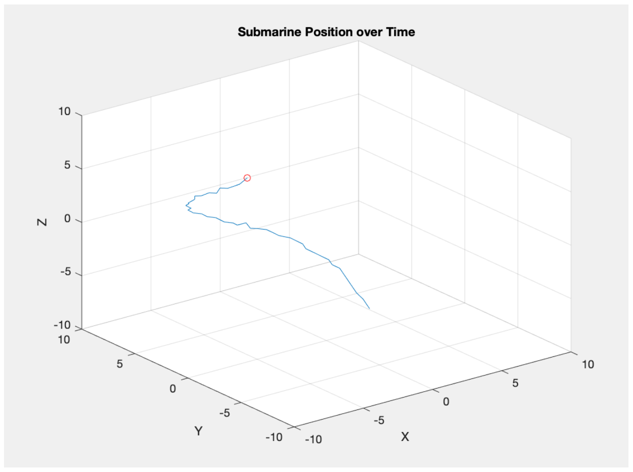

Figure 5: The trajectory of the submarine through time. The red circle is the most recent measurement.

5 Summary and Conclusions

Given noisy acoustic data for `262144` locations in space over `49` different measurements, we were tasked with finding a submarine navigating this space. We were able to isolate the signature frequency emitted by the submarine, which we then used to filter the data in Fourier space. Once filtered, the locations of the submarine were easily identified, which we used to track the path of the submarine through the time measured.

Our particular utilization of Fourier space highlights the phenomena described by the Fourier Uncertainty Principle. For example, while the submarine moved through the volume with time, each sample retained the same signature frequency, allowing us to successfully extract the correct frequency through averaging. However, this same notion made it difficult to retroactively locate the submarine in space. While the methods we utilized are simple, the understanding and application of the Fourier Transform continue remains challenging,

Appendix A MATLAB Functions

- `y = linspace(x1,x2,n)` returns a row vector of `n` evenly spaced points between `x1` and `x2`.
- `[X,Y] = meshgrid(x,y)` returns 2-D grid coordinates based on the coordinates contained in the vectors `x` and `y`. X is a matrix where each row is a copy of `x`, and `Y` is a matrix where each column is a copy of `y`. The grid represented by the coordinates `X` and `Y` has `length(y)` rows and `length(x)` columns.
- `reshape(data,x,y,z,t)` returns a new array with size `(x, y, z, t)` which is filled with the same elements as `data`.
- `fftn(data)` returns the Discrete Fourier Transform of `data` in the same dimension as the inputted `data`.
- `fftshift(dataf)` returns the elements of `dataf` rearranged to undo the shifted indices of the Fast Fourier Transform algorithm. This is accomplished on every dimension of `dataf`.
- `ifftn(dataf)` returns the inverse of Discrete Fourier Transform of `dataf` in the same dimension as the inputted spectral `dataf`
- `ifftshift(dataf)` returns the elements of `dataf` rearranged to align with the shifted indices of the Fast Fourier Transform algorithm. This is accomplished on every dimension of `dataf`.

- `[M,I] = max(dat, [], 'all')` returns the maximum value `M` of all elements of `dat`. The index of the maximum value amongst the entirety of `dat` is stored at `I`.
- `[x, y, z] = ind2sub(size, I)` returns the coordinates corresponding to the single index for the `size` of a given object.

Appendix B MATLAB Code

```
clear all; close all; clc;
load("subdata.mat");

L = 10; % spatial domain
n = 64; % Fourier modes
realizations = 49; % number of samples in time

% create discretized grid of spacial domain
x2 = linspace(-L,L,n+1);
x = x2(1:n);
y = x;
z = x;
[X,Y,Z]=meshgrid(x,y,z);

% create discretized grid of spectral domain
k = (2*pi/(2*L))*[0:(n/2 - 1) -n/2:-1];
ks = fftshift(k);
[Kx,Ky,Kz]=meshgrid(ks,ks,ks);

% reorganize data into 49 realizations of 64x64x64 volume
dat(:,:,:,:)=reshape(subdata,n,n,n, realizations);
```

Listing 1: Example code from external file.

```
% ------------------------PART 1---------------------------

% sum each realization in fourier space,
ave = zeros(n,n,n);
for i = 1:realizations
    datf = fftn(dat(:,:,:,i));
    ave = ave + datf;
end
ave = abs(fftshift(ave)/realizations);

% find coords of max value, represents center frequency
[M, I] = max(ave(:));
[x0, y0, z0] = ind2sub(size(ave),I);

% organize center frequency
Kx0 = Kx(x0, y0, z0)
Ky0 = Ky(x0, y0, z0)
Kz0 = Kz(x0, y0, z0)

% plot of slice of frequency noise (t=1, z = z0)
figure(1)
surf(ks,ks, abs(datf(:,:,z0) / max(abs(datf(:,:,z0)), [], 'all')));
axis([-L L -L L 0 1]);
title('Slice of Frequency Magnitude (49th measurement)');
xlabel('Kx');
ylabel('Ky');
zlabel('Magnitude');

% plot of averaged frequency data
figure(2)
surf(ks,ks, ave(:,:,z0)/M);
axis([-L L -L L 0 1]);
title('Magnitude of Averaged Frequencies at Center Frequency');
xlabel('Kx');
ylabel('Ky');
zlabel('Magnitude');
```

```
% ------------------------PART 2 & 3---------------------------

% define filter function (Gaussian)
tau = .2;
g = exp(-tau* ((Kx-Kx0).^2 + (Ky-Ky0).^2 + (Kz-Kz0).^2));

% filter frequencies around center frequency
% then return to time domain, find coords of max
coords = zeros(49, 3);
for i = 1:realizations
    datf = fftshift(fftn(dat(:,:,:,i)));
    fdatf = g .* datf;
    fdat = ifftn(ifftshift(fdatf));

    [M, I] = max(fdat,[],'all', 'linear');
    [xi, yi, zi] = ind2sub(size(ave),I);
    coords(i, :) = [X(xi,yi,zi) Y(xi,yi,zi) Z(xi,yi,zi)];
end

% plot of averaged frequency data
figure(3)
surf(x,y, abs(dat(:,:,zi)));
axis([-L L -L L 0 1]);
title('Slice of Unfiltered Data in Time Domain');
xlabel('X');
ylabel('Y');
zlabel('Magnitude');


xi = coords(:,1);
yi = coords(:,2);
zi = coords(:,3);

% plot path of submarine
figure(4)
plot3(coords(:,1), coords(:,2), coords(:,3)); hold on;
plot3(coords(end,1), coords(end,2), coords(end,3), 'ro');
axis([-L L -L L -L L]), grid on
title('Submarine Position over Time');
xlabel('X');
ylabel('Y');
zlabel('Z');
```