AMATH 482 - Time Frequency Analysis of Submarine

Johnny Van

January 24, 2020

Abstract

We performed time-frequency analysis on noisy acoustic data from the Puget Sound to detect a moving submarine emitting an unknown frequency with algorithms emphasizing the Fourier Transform. We analyzed the data by averaging the spectrum of raw acoustic data which allows us to identify it's distinctive frequency signature in which a filter created based on this signature. Then we apply the filter to the data to effectively isolate this frequency in order to approximately find the submarine's trajectory in real space and time.

1 Introduction and Overview

We are given broad spectrum of acoustic data taken over 24 hours in 30 minute increments that was taken from the Puget Sound. A moving submarine is equipped with new technology that is emitting a signal with an unknown frequency signature. Using primarily MATLAB, this project focuses on trying to analyze this data and find the submarine's coordinates by accessing the data's frequency information with the Fourier Transform, and using algorithms based on averaging and filtering signals to deduce the submarine's approximate location. The project's main objectives are:

1. Through averaging of the spectrum, determine the frequency signature (center frequency) generated by the submarine.
2. Filter the data around the center frequency determined above in order to denoise the data and determine the path of the submarine. Plot the path of the submarine.
3. Where should you send your P-8 Poseidon subtracking aircraft?

2 Theoretical Background

If we are given an audio signal and plot it as amplitude of the signal vs. time, is there a way to break it down into its frequencies? Well if we were given a function of signal vs. time, we would break it down into sines and cosines of different frequencies. First let's mention Euler's formula,

$$e^{i,\theta} = cos(\theta) + isin(\theta) \tag{1}$$

Using Euler's formula we can convert formulas that utilize sines and cosines into formulas with complex exponentials back and forth.
Then, by the textbook, [1], we are introduced to the Fourier Transform. Suppose we are given a function f(x), we can define the Fourier Transform of f(x) as $\hat{f}(k)$ and its formula is defined by Equation (2).

$$\hat{f}(k) = \frac{1}{\sqrt{2\pi}} \int f(x) e^{-ikx} dx \tag{2}$$

The Fourier Transform takes a function of space/time and converts it into a function of frequencies. Then, if instead you are given $\hat{f}(k)$ and want to find f(x), then you can utilize the inverse Fourier transform to do so, and effectively changing the function's domain back to the space/time instead of frequency. The Inverse Fourier Transform is given by Equation (3).

$$f(x) = \frac{1}{\sqrt{2\pi}} \int \hat{f}(k) e^{ikx} dx \tag{3}$$

From Euler's formula, since we know that e^{ikx} is similar to sin(kx) and cos(kx), the value of k determines the frequences of these sine and cosine waves. So, we can see that the Fourier Transform takes a function of space(or time), x, and converts it to a function of k, or frequency.

2.0.1 Discrete Fourier Transform

In general the Fourier transform is not useful for analyzing data like audio clips because the fourier transform requires an infinite domain. This is where the Discrete Fourier Transform (DFT) comes into play.

Usually when we are given data, we are given a discrete set of points, perhaps something like an signal received every 30 minutes. However it is important to note that between these points, we will never be able to clearly capture the extremely high frequencies, so we would not expect to find information about a signal with large frequency.

Now suppose we receive a sequence of N values that are values sampled at equally-spaced points. Then, discrete fourier transform is given by the formula,

$$\hat{x}_k = \frac{1}{N} \sum x_n e^{\frac{2\pi i k n}{N}} \tag{4}$$

2.0.2 Fast Fourier Transform

The DFT has its advantages, but the Fast Fourier Transform improves upon it. Not only does the fast fourier transform operate faster when implemented in MATLAB, the FFT has a total complexity of $O(Nlog(N)$ which is significantly lower than the total complexity of DFT when processing large amounts of data.

The algorithmn of the Fast Fourier Transform basically is splitting the sequence of N values into half, so instead of performing DFT on the whole sequence, we would perform 2 DFTs on a sequence of length $\frac{N}{2}$. However, we keep repeating this process forever. Luckily, we do not have to calculate this by hand as MATLAB has a built-in function to perform the Fast Fourier Transform.

Note: It is important to understand that the Fourier Transform can transform functions/sets of data in 2D and 3D. When we perform the fourier transform on 1D data, it decomposes the frequency in one direction. Performing the transformation on 2D would decompose the signal into frequencies into 2 directions, and so on. In this project, the data from the submarine is 3D as it represents spatial coordinates in the cartesian plane.

2.0.3 Filtering Noisy Data

In the real world, we receive data that is most likely very noisy, and even if we perform a Fourier Transform on the raw data, it is still very difficult to interpret. Suppose that we do know the exact frequency of the signal we are seeking, then it is actually possible to analyze noisy data to try to detect a signal.

In order to do this, we must filter our data by applying a spectral filter around this known frequency in order to effectively de-noise much of our data. Mathematically, to apply this filter, which is a function, we would just multiply it by the signal (in frequency domain). There are many filters to use, but a rather simple one is the Gaussian function.

$$F(k) = e^{-\tau(k-k_0)^2} \tag{5}$$

The parameter τ represents the width of the filter, and the constant k_0 represents the center of the filter, but most importantly it represents the exact frequency signature of the signal.

2.0.4 Averaging over Many Realizations

Filtering the data is an effective tool for reducing noise but it fails to take advantage of two facts.The noise is white and that the signal will keep being emitted over time, perhaps at different intervals. It is known that white noise has zero mean, so if you average over many realizations (in frequency space), the noise from each realization will cancel out and the unqiue frequency we are looking for will begin to isolate itself. Generally, the more realizations you average over, the more our averaged signal will look like the true signal we are looking for.

2.0.5 Limitations of the Fourier Transform

There are some caveats to the Fourier transform. The raw data we receive tells us everything about space and time, but nothing about frequency information. When we perform a Fourier transform on the data, we receive information about whether or not a specific frequency exists, but it does not tell when this frequency occurs. So whenever we are playing with the data, we are always in either the space/time domain or the frequency domain, so we back and forth in order to uncover information about both. Using the Fourier Transform works great with periodic signals, but doesn't work well with non-stationary signals. A technique that gives information about both time and frequency is the Gabor transform, but that's a topic for another paper and not necessary for this project.

3 Algorithm Implementation and Development

3.0.1 Time-Frequency Analysis Preparation

It is important to understand that the data we are given, is given to us as a 262144x49 which is a condensed form. In order to actually use the data, we must reshape the 262144 values into a 64x64x64 matrix at each of the 49 realizations which accurately shows locations of the noisy signal in the Cartesian space. The data we are working with is better represented as a 64x64x64x49 matrix.

Before we implement our algorithmn on the data, we must first set up our spatial and frequency domains. Our spatial domain is set to be [-10,10] in each direction since most of the raw data lies in this box. And since we have 64 data points in each direction, our fourier modes is equal to 64. We create our frequency domain based on the number of fourier modes, and since Matlab's fft function assumes frequency is a 2π periodic function, we must rescale frequency domain by $\frac{2\pi}{2L}$ to account for this and also shift the frequency array using fftshift to arrange it in the correct order. Then a meshgrid of the space/time and frequency domain must be created since our data is in 3-D in space domain, so the Fourier Transform of the data will be in 3-D as well in the frequency domain.

Note: Due to matlab's implementation of the Fast Fourier Transform function(fftn), everytime we call the function, the transformed data, is arranged in a sort of backwards order which does not align with the frequency domain we created. So everytime we use the 3-D Fourier Transform, we must perform a shift on the transformed data to align it with our frequency domain.

3.0.2 Algorithm 1 : Averaging over Many Realizations and Finding Frequency Signature

Our first objective is averaging the spectrum and find the center frequency signature. To average the raw acousitc data, first we perform the Fourier Transform on the data at each realization and average the data up over all the realizations, effectively cancelling out most of the white noise. Then the frequency signature of the submarine is the frequency with the largest magnitude in the averaged data since most of the noise has been cancelled out and only the submarine's signal should remain. This is the algorithm is what I used for finding the center frequency.

1. Reshape the data into it's appropriate form (64x64x64) at a single realization.
2. Perform the 3-D Fourier Transform on the data.
3. Perform multiple iterations of the above steps to account for all realizations and sum up all the transformed data.
4. Take the absolute value and perform a fftshift on the sum, then divide this result by the total number of realizations of your data to cancel out the white noise.
5. The frequency signature of the submarine will be the frequency with the maximum magnitude in the averaged data. Find the max of the averaged data and get it's linear index.
6. Using the linear index, find the actual multidimensional indexes of the submarine's frequency signature.

7. Using the indexes of the frequency signature, plug them into Kx,Ky,Kz, the meshgrids in frequency domain, respectively to find the frequency of the signal with respect to each direction.

3.0.3 Creating the 3-D Gaussian Filter

After you find the center frequency signature, the next step is to create the Gaussian filter. Since our data is 3-D, this means we must create a 3-D Gaussian filter by multiplying 3 Gaussian functions together, each representing a filter function in one of three directions in the frequency domain.

$$\text{filter} = e^{-\tau(\text{Kx - kx.center})} * e^{-\tau(\text{Ky - ky.center})} * e^{-\tau(\text{Kz - kz.center})}$$

3.0.4 Algorithm 2: Filtering the Data and Finding Submarine Location

Now that we have created our filter, we can effectively denoise most of the data around our center frequency at each realization, and if we take the inverse 3-D Fourier Transform on our transformed data, the maximum of this reverted data will be where the submarine is in the space domain. It is important to note that instead of shifting the data after applying the 3-D transform, I decided to shift the filter instead, which delivers the same result.The algorithm I used to find the approximate location of the submarine at each realization is below:

1. Reshape the data into it's appropriate form.
2. Perform the 3-D Fourier Transform on the data.
3. Multiply the transformed data by the shifted gaussian filter.
4. Perform a inverse 3-D fourier transform on the filtered data, reverting the data back to the space domain.
5. Find the max of your reverted data and find it's linear index. Then using the linear index, find the actual multidimensional indexes of the submarine's location.
6. Using the indexes of the max, find the approximate location of the submarine with respect to x, y, and z.
7. Perform multiple iterations of the above steps to account for all 49 realizations, and record the submarine's x,y, and z coordinates at each realization.

4 Computational Results

4.0.1 The Center Frequency of the Submarine

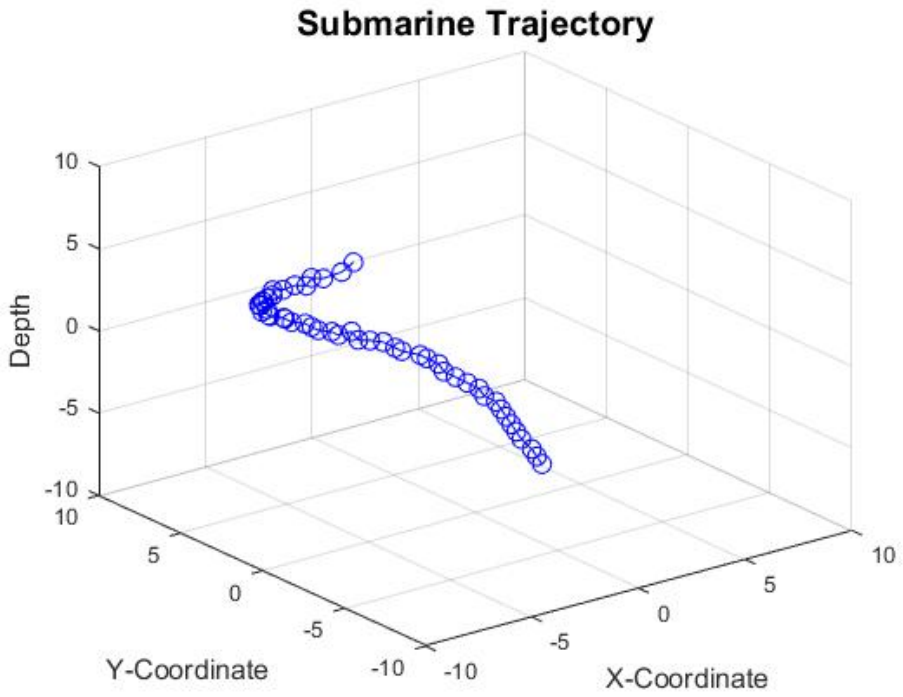
After using Algorithm 1, I have found that the unknown center frequency signature of the submarine is:

$$(5.3407, -6.9115, \ \ 2.1991)$$

Each of the three numbers correspond to the frequency in the Kx, Ky, and Kz direction respectively.

4.0.2 The Trajectory of the Submarine

Now that we know the frequency signature of the submarine, we created a 3-D filter based off the Gaussian function, and used Algorithm 2 to compute the approximate location of the submarine at each of the 49 realizations. After storing the locations of the submarine, I plotted the submarine's trajectory in Matlab in Figure 1. The submarine first started off rather deep in the ocean, then as 24 hours past, it slowly swam up towards the ocean surface. Then Tables 1 and 2 contains the x and y coordinates (Earth's surface) of the submarine at each of the 49 realizations, so if a helicopter were to track it from above the ocean, it should be at these coordinates at the designated times.



: Figure 1: Trajectory of Submarine

Realization(j)	X Position	Y Position
1	3.125	0
2	3.125	0.3125
3	3.125	0.625
4	3.125	1.25
5	3.125	1.5625
6	3.125	1.875
7	3.125	2.1875
8	3.125	2.5
9	3.125	2.8125
10	2.8125	3.125
11	2.8125	3.4375
12	2.5	3.75
13	2.1875	4.0625
14	1.875	4.375
15	1.875	4.6875
16	1.5625	5
17	1.25	5
18	0.625	5.3125
19	0.3125	5.3125
20	0	5.625
21	-0.625	5.625
22	-0.9375	5.9375
23	-1.25	5.9375
24	-1.875	5.9375
25	-2.1875	5.9375

Table 1: X and Y Coordinates of Submarine

Realization(j)	X Position	y Position
26	-2.8125	5.9375
27	-3.125	5.9375
28	-3.4375	5.9375
29	-4.0625	5.9375
30	-4.375	5.9375
31	-4.6875	5.625
32	-5.3125	5.625
33	-5.625	5.3125
34	-5.9375	5.3125
35	-5.9375	5
36	-6.25	5
37	-6.5625	4.6875
38	-6.5625	4.375
39	-6.875	4.0625
40	-6.875	3.75
41	-6.875	3.4375
42	-6.875	3.4375
43	-6.875	2.8125
44	-6.5625	2.5
45	-6.25	2.1875
46	-6.25	1.875
47	-5.9375	1.5625
48	-5.3125	1.25
49	-5	0.9375
-	-	-

Table 2: X and Y Coordinates of Submarine

5 Summary and Conclusions

Given spatial acoustic data of the Puget Sound, we are able to deduce approximately where the submarine is at all times by averaging the data and finding the frequency with the highest magnitude to find the submarine's unique frequency signature. Then using the frequency signature, we can filter the data to uncover the submarine's approximate location at each of the 49 realizations. We were able to find out that the submarine was initially deep in the ocean when data was first being collected, and about 24 hours later, the submarine was near the ocean surface. This is truly a remarkable result as the initial data we received was full of noise, and uncovering the submarine's frequency signature would be impossible without methods like the Fourier Transform.

In conclusion, this sort of application can be used for a wide variety of scenarios such as a signal from an incoming airplane or identifying unique notes with their distinctive frequencies in a sample of music. If we have enough realizations of this raw data, then we are able to average the signal in order to find it's unique frequency signature. We create a filter based off it's frequency signature, we can effectively denoise the data and be able to use the inverse Fourier Transform to find out where the signal originates or when it occurred.

References

[1] Jose Nathan Kutz. *Data-driven modeling & scientific computation: methods for complex systems & big data*. Oxford University Press, 2013.

Appendix A MATLAB Functions

Add your important MATLAB functions here with a brief implementation explanation. This is how to make an **unordered** list:

- `y = linspace(x1,x2,n)` returns a row vector of `n` evenly spaced points between `x1` and `x2`.
- `[X,Y,Z] = meshgrid(x,y,z)` returns 3-D grid coordinates defined by the vectors x, y, and z. The grid represented by X, Y, and Z has size length(y)-by-length(x)-by-length(z).X is a matrix where each row is a copy of x, Y is a matrix where each column is a copy of y, and Z is a matrix where each layer is a copy of z.
- `Y = fftn(X)` returns the 3-D Fourier Transform of a function X.
- `Y = fftshift(X)` returns the shifted form of a matrix X.
- `Y = ifftn(X)` returns the inverse 3-D Fourier Transform of a matrix X.
- `[Max, Index] = max(x(:))` returns the max of an array of values and gives its linear index.
- `[`I_1, I_2, I_3`] = ind2sub(sz, linearIndex)` returns I_1, I_2, I_3 which represents the equivalent multidimensional subscripts corresponding to the linearIndex for a multidimensional array of size sz.
- `plot3(X,Y,Z)` returns a plot using these coordinates in 3-D space.

Appendix B MATLAB Code

```
%% Part 1 Creating the Space/Time and Frequency Domains
load subdata.mat % Imports the data as the 262144x49 (space by time) matrix called subdata

L = 10; % spatial domain
n = 64; % Fourier modes

x2 = linspace(-L,L,n+1); x = x2(1:n); y =x; z = x; % Creates spatial domain axes
k = (2*pi/(2*L))*[0:(n/2 - 1) -n/2:-1]; ks = fftshift(k); % Frequency domain scaled by 2*pi/2L

[X,Y,Z]=meshgrid(x,y,z); % Creates a grid of the spatial domain
[Kx,Ky,Kz]=meshgrid(ks,ks,ks); % Creates a grid of frequency domain

Utn_ave = zeros(n,n,n);

%% Part 2 Averaging the Spectrum and Finding Frequency Signature
for j=1:49
    Un(:,:,:)=reshape(subdata(:,j),n,n,n); % Reshapes data into 64x64x64 matrix at realization j
    Utn = fftn(Un);  % Computes 3-D Fast Fourier Transform on data.
    Utn_ave = Utn_ave + Utn; % Sums transformed data
end

Utn_ave = abs(fftshift(Utn_ave))/49; % fftshift to rearrange data into appropiate order.

[Max_ave, Max_index] = max(abs(Utn_ave(:))); % Max_Ave = largest magnitude, Max_index = linear Index
                                             % of frequency corresponding to Max_Ave
[X_index, Y_index, Z_index] = ind2sub([n,n,n],Max_index); % Multidimensional indexes of the max

center_Kx = Kx(X_index,Y_index,Z_index); % Freq Signature with respect to Kx direction
center_Ky = Ky(X_index,Y_index,Z_index); % Freq Signature with respect to Ky direction
center_Kz = Kz(X_index,Y_index,Z_index); % Freq Signature with respect to Kz direction

%% Part 3 Applying the Filter and Finding Position of Submarine

tau = 0.2; % Width of filter
filter = fftshift(exp(-tau*((Kx-center_Kx).^2 +(Ky-center_Ky).^2 +(Kz-center_Kz).^2)));

for j=1:49
    Un(:,:,:)=reshape(subdata(:,j),n,n,n); % Reshapes data into 64x64x64 matrix
    Utn = fftn(Un); % 3-D Fourier Transform on data
    Unft = filter.*Utn; % Apply filter to transformed data
    Unf = ifftn(Unft); % 3-D Inverse Fourier Transform to go back to space domain

    [Max_Unf, pos_index] = max(abs(Unf(:))); % Finds linear index of largest magnitude of Unf
    [X_index, Y_index, Z_index] = ind2sub([n,n,n],pos_index);

    x_pos(j) = X(X_index, Y_index, Z_index); % X_coordinate of Submarine
    y_pos(j) = Y(X_index, Y_index, Z_index); % Y_coordinate of Submarine
    z_pos(j) = Z(X_index, Y_index, Z_index); % Z_coordinate of Submarine

end
```

```
plot3(x_pos, y_pos, z_pos,'-o','Color','b','MarkerSize',8) % Plots Submarine Trajectory in 3-D
axis([-10 10 -10 10 -10 10]), grid on,
xlabel('X-Coordinate', 'FontSize', 12)
ylabel('Y-Coordinate', 'FontSize', 12)
zlabel('Depth', 'FontSize', 12)
title('Submarine Trajectory', 'FontSize', 15)
drawnow
hold on
```