AMATH 482 Homework 1

Jonah Wu

January 27, 2021

Abstract

We are searching for a submarine in the Puget Sound. We can detect noisy acoustic data from an unknown acoustic frequency emitted by the submarine. First, We will use the noisy acoustic data to find the frequency signature (center frequency) generated by the submarine. Next, we will filter the data around the center frequency we find to denoise the data and determine the path of the submarine.

1 Introduction and Overview

To find the trajectory and location of the submarine, we collect acoustic data from the acoustic frequency emitted by the submarine. However, the data we collect would not be as clean and perfect as we want them to be. In this project, our frequency data contains white noises that have a mean of 0. So, we can get rid of those noises by averaging the data in the frequency domain in order to find the center frequency. Next, after finding the center frequency, we know what frequency we want to look for, so we can focus on that frequency by filtering the acoustic data. Finally, after we filter the noises out, we then can determine the trajectory and the final location of the submarine. Using MATLAB, we are going to use Fourier Transform and averaging to look for the center frequency and apply Gaussian Filtering to filter out the noise, focusing on only the submarine's center frequency that we find.

2 Theoretical Background

2.1 Fourier Transform

The Fourier Transform breaks down a function or signal of waves into an alternate representation of sines and cosines, showing that we can rewrite any waveform as the sum of sinusoidal functions. Suppose we have a function $f(x)$ with $x \in \mathbb{R}$, the Fourier Transform is defined as:

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

In addition, the Fourier Transform has an inverse and is defined as:

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

In summary, the Fourier Transform transforms a function of space or time, x, into a function of frequencies, k. However, the above Fourier Transform is an integral over infinity, which is not feasible for our computing software such as MATLAB. Instead, we will be using the Fast Fourier Transform for transforming data over a finite interval. In addition, when performing over a large data, Fast Fourier Transform is much faster than the original Fourier Transform. Other differences between the Fast Fourier Transform and the original one are that Fast Fourier Transform centers a frequency of 0 and that it works in a domain of period 2π. As a result, we must re-scale the frequencies by $2\pi/2L$ since the Fast Fourier Transform assumes 2π periodic signals.

2.2 Signal Averaging for Center Frequency

In discrete time, white noise is a discrete signal that has a sequence of random variables with 0 mean. Thus, since we have white noise as our noise in this case, which it affects all frequencies in the same way, we can reduce the white noise by taking the average of the data which we transformed into the frequency domain. Next, in the frequency domain, while white noise has a mean of 0 and a frequency emitted by the submarine actually exists, we can then find the center frequency by looking for which frequency does the spike appear at after signal averaging since the noises are cancelled out.

2.3 Gaussian Filtering

After we have the center frequency, which is the frequency emitted by the submarine, we can ignore all the signals of frequencies far from the center frequency. We can achieve this filter by simply multiplying our data with a Gaussian function with a mean of the desired center frequency since we want to ignore the signals or noises far from this frequency, and this Gaussian function is defined as:

$$F(k) = e^{-\tau(k-k_0)^2} \tag{3}$$

where k_0 is the center frequency that determines where the filter should focus on, and τ is a value that decides how wide the filter should be. Overall, after we apply this filter by multiplying it with the data in the Fourier domain, we can get rid of the noise from the original data by transforming the product back into the time domain.

3 Algorithm Implementation and Development

We will be combining the Fourier Transform and the signal averaging for center frequency into algorithm 1 (step 1 - 3), and have Gaussian Filtering as algorithm 2 (step 1, 2, 4 - 6). The MATLAB code for this part can be found in **Appendix B**, and **Appendix A** explains the MATLAB functions that are used.

General break down:

1. Loads in the noisy acoustic data and reshapes it for following computations.
2. Transforms the noisy data into the frequency domain using Fast Fourier Transform.
3. Finds the center frequency by taking the average of the transformed data over 49 realizations.
4. Filters out the noise by multiply a Gaussian function with k_0 as the center frequency to the noisy data.
5. Transforms the filtered data back into the space and time domain.
6. Finds the trajectory and the final location of the submarine by recording the coordinates using the filtered data.

Algorithm 1: Fast Fourier Transform + Signal Averaging for Center Frequency

```
Import data from subdata.mat
Initialize total as 0
for $j = 1 : 49$ do
    Reshape data at time = $j$ into $64 \times 64 \times 64$ matrix
    Perform Fast Fourier Transform on data using fftn() and fftshift() and add it to total
end for
Find the average by dividing the absolute value of the total by 49
Find the index that has the maximum of the averaged signal using ind2sub() in order to find the center
frequency
```

Before loading in the data, we initialize the Fourier mode with $n = 64$ since the data has 49 columns of 262144 (64^3) rows. Next, the spatial domain is defined to be in the interval of $(-L, L)$ with a length of $2L$, in which L is 10 in this case, so we can define our vectors to be ranged from $-L$ to L with n values with upper-bound L excluded. We then scale the points by $2\pi/2L$ since Fast Fourier Transform assumes 2π periodic signals. Finally, we can find the center frequency by looking for which k has the max value after averaging the signals.

Algorithm 2: Gaussian Filtering for Trajectory

```
Define the Gaussian function as filter
Initialize coordinates to save trajectory
for $j = 1 : 49$ do
    Reshape data at time = $j$ into $64 \times 64 \times 64$ matrix
    Perform Fast Fourier Transform on data using fftn() and fftshift()
    Filter out the noise by multiplying filter to the transformed data
    Transform the filtered data back to spatial domain using ifftn()
    Find and save the location of the submarine at time = $j$ to coordinates using the filtered data
end for
Plot the path of the submarine using plot3()
```

Finally, we then can specify the Gaussian Filtering with the the center frequency we get from Algorithm 1. τ determines how wide the filter should be. Larger τ indicates that we filter out less noise while a very tiny τ indicates that the filter is super thin. 0.05 was found to be small enough because the final location of the submarine converges when τ is 0.05 as τ gets smaller. Thus, I set τ as 0.05 in my computation.

4 Computational Results

4.1 Frequency Signature

After performing Algorithm 1, we find out that the frequency signature, which is the center frequency, is

$$[k_{0x}, k_{0y}, k_{0z}] = [5.3407, -6.9115, 2.1991]$$

4.2 Submarine Path

After performing Algorithm 2, we can plot the trajectory of the submarine with the coordinates we found in Figure 1:

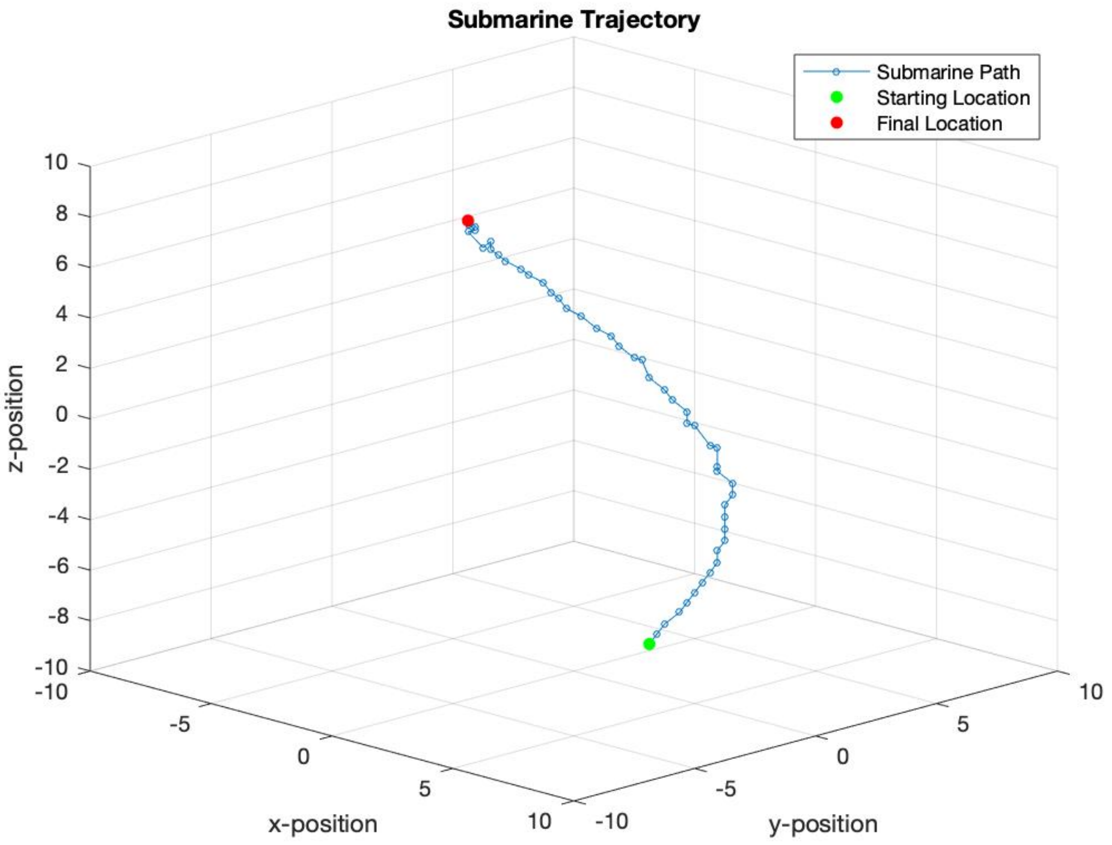

Figure 1: The path of the submarine

4.3 Submarine Coordinates

After performing Algorithm 2, we can find the final location of the submarine by simply taking the values of the last row of the `coordinates`, which is going to be (-5.3125, 0.9375, 6.25), so our final (x, y) coordinate of the submarine is (-5.3125, 0.9375). In addition, we can determine where to send P-8 Poseidon sub-tracking aircraft at any given time with Table 1 of (x, y) coordinates:

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

Table 1: (x, y) coordinates of the submarine at $t = j$

5 Summary and Conclusions

In conclusion, averaging the signals in the Fourier domain can be used to reduce the white noise from a real world wave signal. A frequency signature (central frequency) can be determined by visualizing where the spike is over the averaged data. In this case, we found out that the frequency signature of the submarine is $[k_{0x}, k_{0y}, k_{0z}] = [5.3407, -6.9115, 2.1991]$. Next, a filter can be applied after we know what frequency to target for using a Gaussian function. Since we are working with the data in the transformed frequency domain, we have to use the inverse Fourier transforming to get back to the spatial domain of our filtered data. Finally, after we reduce the noise from the original noisy acoustic data, the data tells us the path of the submarine at given time, and its final location at (-5.3125, 0.9375, 6.25).

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.
- `Y` = `abs(X)`: returns the absolute value of `X`.
- `Y` = `exp(X)`: returns the exponential e^x for each element in array `X`.
- `M` = `max(A)` returns the maximum elements of an array.
- `sz` = `size(A)`: returns a row vector whose elements are the lengths of the corresponding dimensions of A.
- `B` = `reshape(A,sz)` reshapes `A` using the size vector, `sz`, to define `size(B)`.
- `X` = `zeros(sz1,...,szN)` returns an `sz1`-by-...-by-`szN` array of zeros where `sz1`,...,`szN` indicate the size of each dimension.
- `Y` = `fftn(X)`: returns the multidimensional Fourier transform of an N-D array using a fast Fourier transform algorithm.
- `Y` = `fftshift(X)`: rearranges a Fourier transform `X` by shifting the zero-frequency component to the center of the array.
- `X` = `ifftn(Y)`: returns the multidimensional discrete inverse Fourier transform of an multi-dimensional array using a fast Fourier transform algorithm.
- [`row`, `col`] = `ind2sub(sz,ind)` returns the arrays `row` and `col` containing the equivalent row and column subscripts corresponding to the linear indices `ind` for a matrix of size `sz`.

Appendix B MATLAB Code

```
clear all; close all; clc

%% Initializing Fourier Domain
L = 10; % spatial domain
n = 64; % Fourier modes
x2 = linspace(-L,L,n+1);
x = x2(1:n);
y = x;
z = x;
k = (2*pi/(2*L))*[0:(n/2 - 1) -n/2:-1];
ks = fftshift(k);
[X,Y,Z] = meshgrid(x,y,z);
[Kx,Ky,Kz] = meshgrid(ks,ks,ks);

%% Algorithm 1: Fast Fourier Transform + Signal Averaging for Center Frequency
load subdata.mat % Imports the data as the 262144x49 (space by time) matrix called subdata
total = zeros(n,n,n);
for j=1:49
    Un(:,:,:) = reshape(subdata(:,j),n,n,n);
    total = total + fftshift(fftn(Un));
end
avg = abs(total)/49;
[¬, index] = max(avg(:));
[i1,j1,k1] = ind2sub(size(avg),index);
k0_x = ks(j1);
```

```
k0_y = ks(i1);
k0_z = ks(k1);

%% Algorithm 2: Gaussian Filtering for Trajectory
tau = 0.05;
filter = exp(-tau*(((Kx-k0_x).^2)+((Ky-k0_y).^2)+((Kz-k0_z).^2)));
coordinates = zeros(49, 3);
for j=1:49
    Un(:,:,:) = reshape(subdata(:,j),n,n,n);
    Un_fft = fftshift(fftn(Un));
    Un_filter = Un_fft .* filter;
    Un_res = ifftn(Un_filter);
    [¬, index] = max(Un_res(:));
    [i2,j2,k2] = ind2sub(size(Un_res),index);
    coordinates(j,1) = X(i2,j2,k2);
    coordinates(j,2) = Y(i2,j2,k2);
    coordinates(j,3) = Z(i2,j2,k2);
end
hold on
plot3(coordinates(:,1),coordinates(:,2),coordinates(:,3),'-o','MarkerSize',3)
plot3(coordinates(1,1),coordinates(1,2),coordinates(1,3),'g.','MarkerSize',15)
plot3(coordinates(end,1),coordinates(end,2),coordinates(end,3),'r.','MarkerSize',15)
title('Submarine Trajectory')
legend('Submarine Path','Starting Location','Final Location')
xlabel('x-position'); ylabel('y-position'); zlabel('z-position')
view(45,20)
axis([-L L -L L -L L]), grid on, drawnow
```