Hanze Zhang
Mr. Jason Bramburger
AMATH 482
Assignment 1
01/27/2021

Abstract: If we have received a set of noisy acoustic signal data from a submarine, how can we locate the submarine and track its position over a 24-hour period? In this assignment, we successfully accomplished the above goal by applying Fast Fourier Transform on the given data along with averaging the signal spectrum and filtering out noisy components from the original signal.

Introduction and Overview:

In this assignment, our noisy acoustic signal data came from a submarine in the Puget Sound. The data contains 262144 rows and 49 columns of measurements and is obtained by using a broad-spectrum recoding of acoustics over a 24-hour period in half-hour increments. Each column is a measurement obtained in a 30-minute time span. Even though we have successfully obtained the data, we do not know the frequency signature of the submarine due to noisy components in the signal. If we cannot determine its frequency signature, we will not be able to determine its location and path.

To solve above problems, we first need to average on the acoustic signal data to find its frequency signature. Then, we can apply a filter on the averaged signal to further denoise the data. Now, we will be able to use Fast Fourier Transform to help us determine locations of the submarine over the 24-hour period in a half-hour increment.

Theoretical Background:

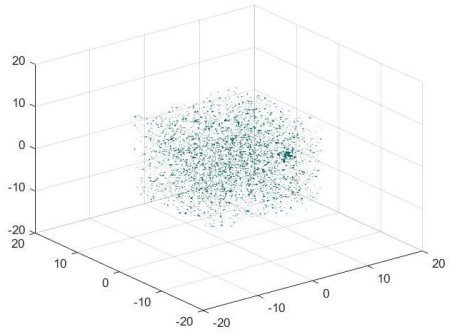

Figure 1 - Noisy Original Data in the frequency domain

It is essentially difficult to determine the frequency signature from a noisy data in the spatial domain. Therefore, we need to use Fast Fourier Transform to transform our data from the spatial domain into the frequency domain. Fast Fourier Transform is an algorithm based on Discrete Fourier Transform. The Discrete Fourier Transform is a sequence of numbers given by:

$$\hat{x}_k = \frac{1}{N}\sum_{n=0}^{N-1} x_n e^{\frac{2\pi i k n}{N}}, k = 0,1,2,\dots,N-1 \qquad (1)$$

In MATLAB implementation, $k = -\frac{N}{2}, -\frac{N}{2}+1, \dots, -1,0,1,2,\dots,\frac{N}{2}-1$ due to aliasing. However, in the Discrete Fourier Transform, calculating each number requires N times operation, which is inefficient when we are working with a large dataset. Thus, we incorporates the idea of divides and conquer into the Discrete Fourier Transform and create the Fast Fourier Transform, which is much more efficient than Discrete Fourier Transform.

Even if we view the data in the frequency domain, represented in Figure 1, we still cannot really distinguish the frequency signature from the rest of frequencies as a lot of them are noise. This is the reason why we have to apply denoise techniques on the given data. In this assignment, we utilize two denoise techniques: averaging and filtering.

After we transform our data from the spatial domain to the frequency domain with Fast Fourier Transform, we can now average on each frequency. Since we expect white noise to have an average of 0, averaging through each frequency can help us cancel out noisy frequencies, making it possible for us to distinguish the frequency of the submarine from the rest of the signal.

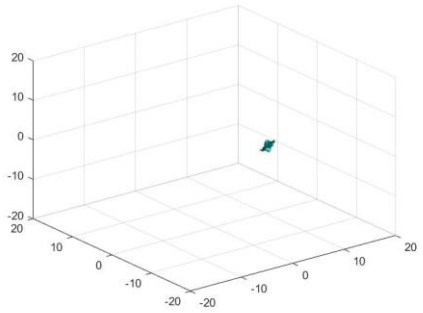

Figure 2 - Averaged Data in the Frequency Domain

On the contrary to averaging, which tries to diminish noisy frequency as much as possible, filtering aims to amplify those frequencies close to the frequency center. At the same time, it also diminishes other frequencies. Thus, filtering can help us further determine the frequency signature of the submarine. However, to utilize a filter, we have to know the approximate place of our target frequency. This is the reason why we use averaging to average out most of white noise first.

After we have successfully performed averaging and filtering, we can then apply Inverse Fast Fourier transform on the filtered data and return to the spatial domain. In this way, we get the original signal without a lot of noise. Then, we will be able to locate the submarine and track its position by looking for the maximum in the spatial domain.

Algorithm Implementation and Development:

Before averaging the data in the frequency domain, we have to import the data first. After we have successfully imported the data, we call the "$reshape$" method in MATLAB to help us transform the data from a 262144 x 49 vector to a 64 x 64 x 64 matrix at each half-hour period. Since averaging in time domain does not provide anything useful, we need to use the "$fftn$" method to transform the data into frequency domain and proceed our analysis there. In frequency domain, we can iterate through our data at each time point and sum each transformed value up. After the iteration, we can divide the summation by 49, which is the number of time periods we have, to obtain the average. After we successfully find the average, we can call the method "max" on our averaged data to find the maximum frequency and call the method "ind2sub" to find three indices of the maximum frequency in our data. These indices indicate where

in the data we should apply our filter on.

Now, we can build up our filter according to the Gaussian function:

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

In the function, τ determines the width of the filter and k_0 determines the center of the filter. Also, k is the rescaled frequency components calculated based on the spatial domain. It is rescaled by $\frac{2\pi}{L}$ since the Fast Fourier Transform assumes 2π periodic signals. In 3-dimension, the above Gaussian function becomes:

$$F(k) = e^{-\tau((k-k_{x0})^2+(k-k_{y0})^2+(k-k_{z0})^2)} \qquad (3)$$

Here, k_{x0}, k_{y0}, k_{z0} are coordinates of the center of frequency we found with three indices above. Since the filter needs to be applied to every value at each time point, we need to iterate through the raw data at each time point and reshape it to a 64 x 64 x 64 matrix again. Different from what we have done above, instead of summing data up, we need to multiply our filter to the data in the frequency domain. At this point, we have successfully applied our filter on the data. Then we need to transform our data back into the time domain with the function "$iffn$". Since multiplying in the frequency domain will also lead to changes in the time domain, after we transform our data back into the spatial domain, we can find the maximum of the signal and save its indices. These indices can help us find corresponding x, y, z coordinates. The place of the maximum signal will be the location of the submarine at that particular time period. Thus, in the iteration, we can save corresponding x, y, z coordinates at each time period into three different 49 x 1vectors. Then, we can use the command "plot3" to plot the movement of the submarine and track its position over time.

After we can track the movement of the submarine, we can send our P-8 Poseidon sub-tracking aircraft above the submarine at each time period.

Computational Results:

After averaging the spectrum, we found the maximum frequency was at the index $[11,50,40]$. With these indices, we found the center of frequency (frequency signature) was $[5.3407, -6.9115, 2.1991]$. With the center of frequency and $\tau = 0.2$, we built

up the filter:

$$F(k) = e^{-0.2((k-5.3407)^2+(k-(-6.9115))^2+(k-2.1991)^2)} \quad (4)$$

The filter enhanced the frequency signature of the submarine inside the filter window while it also diminished other frequencies outside the filter window. After we transformed the filtered data back into the spatial domain, we recorded indices of the maximum signal and found the position of the submarine at each time point. Figure 3 illustrated the movement of the submarine:

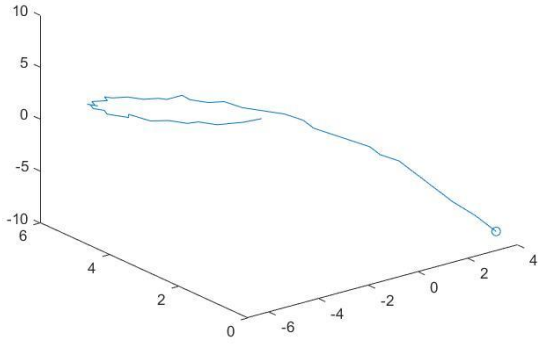

Figure 3 - Movement of the submarine

In the above figure, the circle indicates the starting position of the submarine, which is $[3.1250, 0, -8.1250]$. As we have x, y, z coordinates of the submarine now, we can send our sub-tracking aircraft right above the submarine. The position of the sub-tracking aircraft is summarized into the below table:

	x	y
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
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 1-Position of Sub-tracking Aircraft

Summary and Conclusions:

In conclusion, averaging and filtering on the noisy raw data were very effective at removing noise from a three-dimensional signal. These techniques, along with Fast Fourier Transform, can help us precisely determine the location of a moving submarine at each time period. In addition, we will also know where we should send the sub-tracking aircraft to track the submarine.

Appendix A:

MATLAB Methods:

- $load$: Load variables from file into workspace.
- $reshape$: Reshape array.
- $fftn$: N-D fast Fourier transform.
- $fftshift$: Shift zero-frequency component to center of spectrum.
- $max(A, [], \_\_, 'linear')$: returns the linear index into A that corresponds to the maximum value in A.
- $ind2sub(sz, ind)$: returns the arrays row, col, page containing the equivalent row, column and page subscripts corresponding to the linear indices ind for a matrix of size sz.
- $ifftn$: Multidimensional inverse fast Fourier transform data set from the frequency domain back to the special domain.
- $plot3$: 3-D point or line plot.

Appendix B (MATLAB Code):

```
% Clean workspace2
clear all; close all; clc

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;
k = (2*pi/(2*L))*[0:(n/2 - 1) -n/2:-1]; % frequency component
ks = fftshift(k);

[X,Y,Z]=meshgrid(x, y, z);
[Kx,Ky,Kz]=meshgrid(ks, ks, ks);

ave = zeros(n,n,n); % summation for average
for j=1:49
    Un(:,:,:)=reshape(subdata(:,j), n, n, n);
    Utn = fftn(Un);
    ave = ave + Utn;
    M = max(abs(Utn), [], 'all');
    % close all, isosurface(X, Y, Z, abs(Un) / M, 0.4)
    % axis([-20 20 -20 20 -20 20])
    % grid on
    % drawnow
    % pause(1)
end

figure(1)
isosurface(X, Y, Z, fftshift(abs(Utn)) / M, 0.5)
axis([-20 20 -20 20 -20 20])
grid on
drawnow

ave = abs(fftshift(ave)) / 49; % average to recover the noisy signal
[maximum, index] = max(ave(:));
[r,c,p] = ind2sub(size(ave),index); % locate the signal

figure(2)
isosurface(Kx, Ky, Kz, ave / max(ave(:)), 0.4)
axis([-20 20 -20 20 -20 20]), grid on, drawnow
```

```
kx0 = Kx(r, c, p);
ky0 = Ky(r, c, p);
kz0 = Kz(r, c, p);

tau = 0.2;
filter = exp(-tau * ((Kx - kx0).^2 + (Ky - ky0).^2 + (Kz - kz0).^2));

x_pos = zeros(49,1);
y_pos = zeros(49,1);
z_pos = zeros(49,1);

for j=1:49
    unft(:,:,:)=reshape(subdata(:,j), n, n, n);
    unft = fftn(unft) .* fftshift(filter); %applying filter
    unft = fftshift(unft);
    unf = ifftn(unft); % transform back to time domain
    [maximum,index] = max(abs(unf(:)));
    [x,y,z] = ind2sub(size(unf),index);
    x_pos(j) = X(x, y, z);
    y_pos(j) = Y(x, y, z);
    z_pos(j) = Z(x, y, z);
end

figure(3)
plot3(x_pos, y_pos, z_pos, '-o', 'MarkerIndices',1)

tracker_pos = [x_pos, y_pos];
```