A Submarine Problem

Jiasui Qin

January 27, 2020

Abstract

Signal detection has always been an important research field in nowaday studies. For most acoustic signals, they are often non-stationary and corrupted by unpredictable noise sources when received by underwater systems. There are a large number of methods being proposed to address this issue. This paper will focus on using averaging and filtering to extract the exact signals.

1 Introduction and Overview

We are trying to capture the trajectory of a submarine that emits an unknown acoustic frequency. Given the noisy data of a broad spectrum recording of acoustics, our goal is to identify the locations and the path of the moving submarine. The data was obtained over a 24-hour period in half-hour increments. To transfer the data into a proper format that can be used later, I reshaped it into a 64×64×64×49 matrix. The first three dimensions represent Cartesian coordinates and the last dimension represent time. Therefore, there are 49 realizations and we need to find its location at each realization.

Our method can be divided into two general steps:
1. Determine the frequency signature (center frequency) generated by the submarine.
2. Filter the data around the center frequency and determine the path of the submarine.

To achieve this, we will apply the idea of averaging as well as filtering.

2 Theoretical Background

2.1 Discrete Fourier Transform (DFT)

Fourier transform can be used to converts a function of time to a function of frequencies. Given a function $f(x)$ with $x \in R$, we define the Fourier transform of $f(x)$ by the formula[4]:

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

We can also use the inverse Fourier transform to transform $\hat{f(k)}$ back to $f(x)$[4]:

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

Since e^{ikx} can be represented as the combination of $sin(kx)$ and $cos(kx)$, the value of k represents the frequencies of sine and cosine waves. So, the Fourier transform helps us transform data in the time domain to the associated data in the frequency domain.

Given a discrete set of signals $\{x_0, x_1, ..., x_{N-1}\}$ in the time domain, we can apply the discrete Fourier transform to transform the data, which is a sequence of numbers given by[4]:

$$\hat{x_k} = \frac{1}{N} \sum_{n=0}^{N-1} x_n \mathrm{e}^{\frac{2\pi ikn}{N}} \tag{3}$$

2.2 Averaging

The general idea behind averaging is that the noise from each realization will cancel out if we average over many realizations in the frequency domain. It helps smooth the noisy signal to make the real signal stand out from the background noise. To get the final result, we add signals from each realization together and divide it by the number of realizations.

2.3 Filtering

Given the noisy data, if we know the center frequency of the signal, we can apply the spectral filter around this frequency to remove undesired frequencies. If we multiply the signal in the frequency domain by the filter function, we can then get the filtered signal with most of the noises removed, which helps us detect the real location of the signal. For our problem, we will apply the following Gaussian function as the filter. τ represents the width of the filter and the constant (k_1, k_2, k_3) represents the center coordinates of the filter.

$$F(k) = \mathrm{e}^{-\tau((k-k_1)^2+(k-k_2)^2+(k-k_3)^2)} \tag{4}$$

3 Algorithm Implementation and Development

3.1 Domain Discretization

We define the length of the spatial domain to be [-10, 10] with 64 Fourier modes. For frequency domain, we need to scale the frequencies by $\frac{2\pi}{L}$ since the fast Fourier transform assumes 2π periodic signals.

3.2 Averaging

To find the center frequency generated by the submarine, we want to first get the average signal over all realizations in the frequency domain. The location of the largest value in the final result then represents the location of the center frequency.

General steps of the algorithm:

1. Transfer the signal in the time domain to the corresponding signal in the frequency domain using fast Fourier transform. The Matlab function is fftn()[1] that helps us do the computation.

2. Add signals together for all realizations and compute the average. Note that we need to shift the FFT using Matlab function fftshift()[2] to transfer the interval from $[-\frac{n}{2}, \frac{n}{2}]$ to $[0, n]$ that the FFT takes as default. n is the number of Fourier modes. We also need to take the absolute value of FFT, this is because the signal is complex value.

3. Find the largest value in the final signal and extract its location.

Algorithm 1: Averaging

Import data from `subdata.mat`
Initialize ave = $64 \times 64 \times 64$ zero matrix
for $j = 1 : 49$ **do**
  un = realization j from `subdata`
  ut = fftn(un)
  ave = ave + ut
end for
ave = abs(fftshift(ave))/49
Find the largest value in 'ave' and its associated location

3.3 Filtering

General steps of the algorithm:

1. Initialize the Gaussian filter using the location of the center frequency computed from 3.2.
2. Apply the filter to each realization in the frequency domain.
3. Transfer the filtered signal back to the spatial domain using Matlab function ifftn()[3] to compute the inverse fast Fourier transform. Note that we also need to take the absolute value of the signal because the signal is complex.
4. Find the largest value in the filtered signal and extract its location.

Algorithm 2: Filtering

Continue from algorithm 2
filter = Gaussian filter with the pre-computed center frequency and $\tau = 0.2$
for $j = 1 : 49$ **do**
  unft = filter.$\times$ (realization j in the frequency domain)
  inverse = abs(ifftn(unft))
  Find the largest value in 'inverse' and its associated location
end for

4 Computational Results

4.1 Center Frequency

After getting the averaged signal in the frequency domain, the largest value is located at index [11, 50, 40]. It's associated coordinates in the frequency domain is [5.34, -6.91, 2.19]. Let's compare the original signal and the averaged signal in the frequency domain.

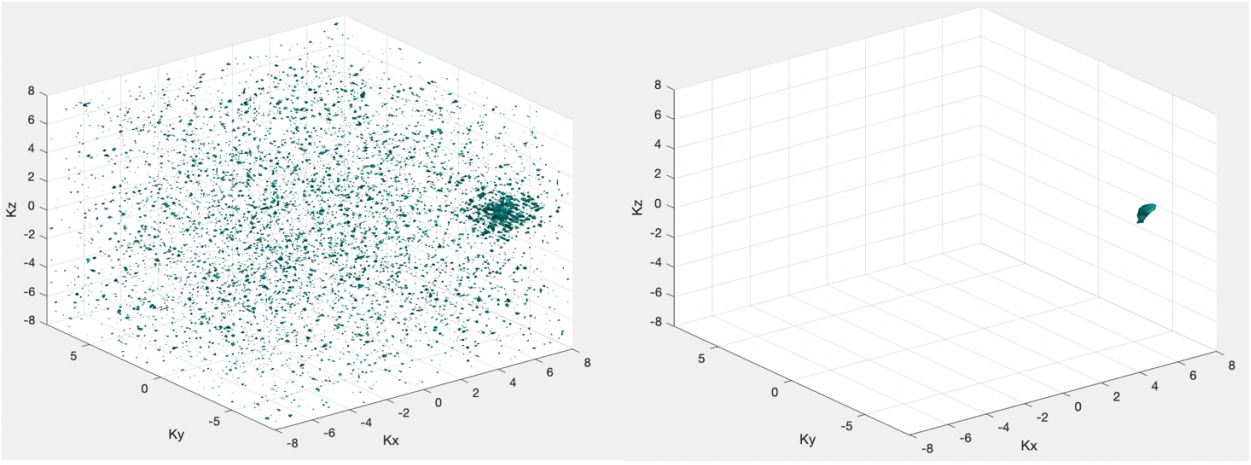

Figure 1: Original signal VS Averaged signal in the frequency domain

4.2 Gaussian Filter

Based on the center frequency we computed, the Gaussian function is then defined as:

$$F(k) = \mathrm{e}^{-0.2((kx-5.34)^2+(ky+6.91)^2+(kz-2.19)^2)} \tag{5}$$

Let's visualize what our filter looks like and what the filtered signal looks like in the frequency domain.

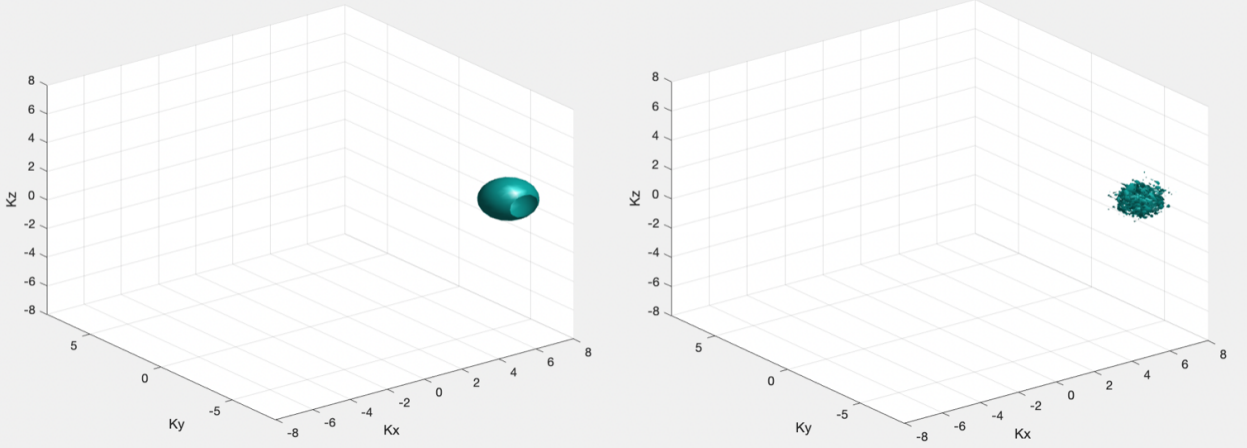

Figure 2: Gaussian Filter VS Filtered signal in the frequency domain

4.3 Submarine Trajectories

After computing the inverse Fourier transform of the filtered signal, we get the following result. Here's the original signal versus the filtered signal in the spatial domain.

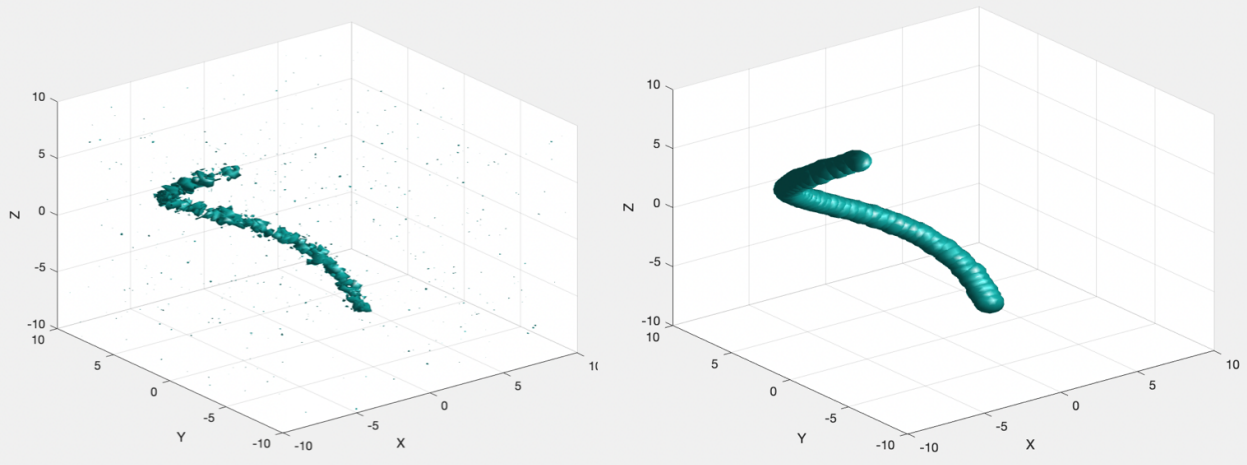

Figure 3: Original signal VS Filtered signal in the spatial domain

The associated coordinate of the maximum value in each filtered signal tells us the submarine's location at different time. Let's see a plot of its trajectory.

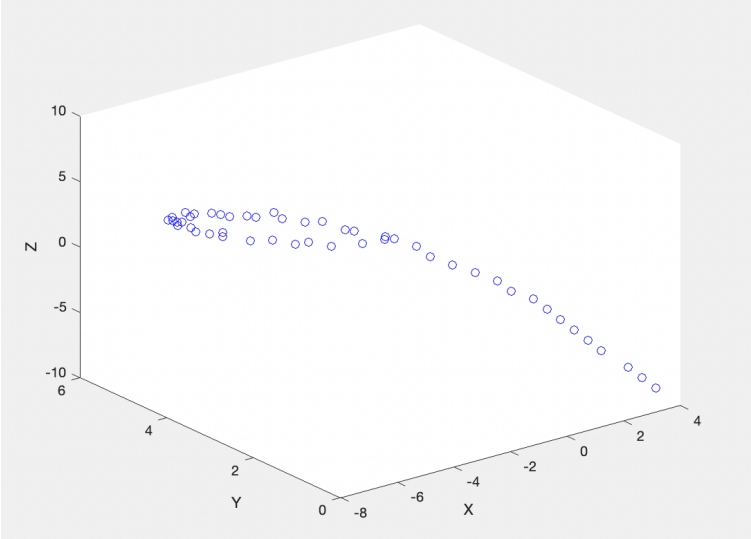

Figure 4: Trajectory of the submarine at different time

If we want to send a subtracking aircraft to track the submarine, we could follow the following path with the x and y coordinates shown in the table below.

time	X	Y	time	X	Y
1	3.125	0	26	-2.8125	5.9375
2	3.125	0.3125	27	-3.125	5.9375
3	3.125	0.625	28	-3.4375	5.9375
4	3.125	1.25	29	-4.0625	5.9375
5	3.125	1.5625	30	-4.375	5.9375
6	3.125	1.875	31	-4.6875	5.625
7	3.125	2.1875	32	-5.3125	5.625
8	3.125	2.5	33	-5.625	5.3125
9	3.125	2.8125	34	-5.9375	5.3125
10	2.8125	3.125	35	-5.9375	5
11	2.8125	3.4375	36	-6.25	5
12	2.5y	3.75	37	-6.5625	4.6875
13	2.1875	4.0625	38	-6.5625	4.375
14	1.875	4.375	39	-6.875	4.0625
15	1.875	4.6875	40	-6.875	3.75
16	1.5625	5	41	-6.875	3.4375
17	1.25	5	42	-6.875	3.4375
18	0.625	5.3125	43	-6.875	2.8125
19	0.3125	5.3125	44	-6.5625	2.5
20	0	5.625	45	-6.25	2.1875
21	-0.625	5.625	46	-6.25	1.875
22	-0.9375	5.9375	47	-5.9375	1.5625
23	-1.25	5.9375	48	-5.3125	1.25
24	-1.875	5.9375	49	-5	0.9375
25	-2.1875	5.9375			

Table 1: x,y coordinates of the submarine at each time

5 Summary and Conclusions

As we can see from the result, applying the technique of averaging helps us localize the center frequency. By using the center frequency, we can then construct a filter that could remove noisy data. These two steps play a significant role in extracting the hidden signal. In conclusion, by taking advantage of the Fourier transform that allows us to transform data between the spatial domain and the frequency domain, we could apply the method of averaging and filtering to make a huge improvement in the field of signal detection.

References

[1] *fftn()*. URL: `https://www.mathworks.com/help/matlab/ref/fftn.html`.

[2] *fftshift()*. URL: `https://www.mathworks.com/help/matlab/ref/fftshift.html`.

[3] *ifftn()*. URL: `https://www.mathworks.com/help/matlab/ref/ifftn.html`.

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

Appendix A MATLAB Functions

- `x2 = linspace(-L/2,L/2,n+1)` returns a row vector of `n+1` evenly spaced points between `-L/2` and `L/2`.
- `[X,Y,Z] = meshgrid(x,y,z)` returns 3-D grid coordinates based on the coordinates contained in the vectors `x`, `y`, and `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 column is a copy of `z`. The grid represented by the coordinates is a $64 \times 64 \times 64$ matrix.
- `zeros(size)` returns a zero matrix with the specified size.
- `fftn(un)` computes the fast Fourier transform of the multidimensional matrix `un`.
- `fftshift(ave)` rearranges a Fourier transform `ave` by shifting the zero-frequency component to the center of the array.
- `ind2sub(size(ave),find(ave == maximum)` returns the indices of the maximum value in the matrix `ave`.
- `ifftn(unft)` returns the multidimensional discrete inverse Fourier transform of `unft` using a fast Fourier transform algorithm.

Appendix B MATLAB Code

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

load subdata.mat % Imports the data as the 262144x49 (space by time) matrix called subdata

L = 20; % spatial domain
n = 64; % Fourier modes
x2 = linspace(-L/2,L/2,n+1); x = x2(1:n); y = x; z = x;
k = (2*pi/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);

% Averaging all realizations to get center frequency
ave = zeros(64, 64, 64);
Ut = zeros(64, 64, 64, 49);
for j = 1:49
    un(:,:,:)=reshape(subdata(:,j),n,n,n);
    ut = fftn(un);
    Ut(:,:,:,j) = ut;
    ave = ave+ut;
end
ave = abs(fftshift(ave))/49;
center = max(ave,[],'all')
[k1_,k2_,k3_] = ind2sub(size(ave),find(ave == max(ave,[],'all')))
k1 = Kx(k1_,k2_,k3_); k2 = Ky(k1_,k2_,k3_); k3 = Kz(k1_,k2_,k3_);

% define a gaussian filter
[kx,ky,kz]=meshgrid(k,k,k);
tau = 0.2;
filter = exp(-tau*((kx - k1).^2+(ky - k2).^2+(kz - k3).^2));

% apply the filter to each realization
unft = zeros(64, 64, 64, 49);
unf = zeros(64, 64, 64, 49);
location = zeros(49,3);
indices = zeros(49,3);
for j = 1:49
    unft(:,:,:,j) = filter.*Ut(:,:,:,j);% apply filter
    inverse = abs(ifftn(unft(:,:,:,j)));% do ifft() to get back to the time domain
    unf(:,:,:,j) = inverse;
    maximum = max(inverse,[],'all');
    [a_,b_,c_] = ind2sub(size(inverse),find(inverse == maximum));
    a = X(a_,b_,c_); b = Y(a_,b_,c_); c = Z(a_,b_,c_);
    location(j,:) = [a,b,c];
    indices(j,:) = [a_,b_,c_];
%     Uncomment code below to visualize the trajectory
%     plot3(a,b,c,'o','Color','b'),hold on
%     xlabel('X')
%     ylabel('Y')
%     zlabel('Z')
%     pause(1)
end
```

Listing 1: Matlab code