Signal Processing

Yukai Yan

January 27th, 2021

Abstract

In this paper, we try to identify the submarine's location by processing the noisy acoustic data. We can achieve this goal by detecting the acoustic frequency it emits. The frequency spectrum that records acoustic data is averaged to reduce the noise and locate the center frequency. A Gaussian function is then used as a filter to remove undesired frequencies and much of the white space. By taking the inverse Fourier transform of filtered data, the path of the submarine can be obtained.

1 Introduction and Overview

1.1 The submarine problem

We are hunting for a submarine in the Puget Sound using noisy acoustic data. It is a new submarine technology that emits an unknown acoustic frequency. Using a board spectrum recording of acoustics, the data is obtained over a 24-hour period in half-hour increments. Note that the size of the data is $64 \times 64 \times 64 \times 48$, where the first three numbers represent the coordinates on the XYZ plane, and the last number represents the time. The submarine is moving, and our goal is to identify the submarine's location and path in this 24-hours period.

1.2 Approach the problem

Unfortunately, given the significant amount of white noise in the data, the frequency that the new submarine emits is indistinguishable to the naked eyes. Therefore, we use the fast Fourier Transform (FFT). The FFT takes a function of time, x, and converts it to a function of frequencies, k. By taking the average of many realizations in the frequency space, the noise from each realization will nearly cancel out, assuming the noise is truly random. We can then use the Gaussian filter around the center frequency to detect the path of the submarine.

1.3 Overview

We will discuss Fourier Transform, averaging, and filtering in Section Two. In Section Three, we will develop the solution and implement the corresponding algorithms. Section Four will include the computational results derived from the FFT approach, as the summary and conclusion will be in Section Five.

2 Theoretical Background

2.1 Fourier Transform

Suppose we are given a function $f(x)$ with $x \in \mathbb{R}$, we can define the Fourier Transform of $f(x)$, written $\hat{f}(k)$ by:

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

We can also recover $f(x)$ using the inverse Fourier Transform:

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

By the Euler's formula, we know that e^{ikx} can be defined by $sin(kx)$ and $cos(kx)$, where the value of k gives the frequency of sine and cosine waves. Therefore, the Fourier transform takes a function of space or time, x, and converts it to a function of frequencies, k.

2.2 Fourier Series

Suppose we are given a function $f(x)$ on the interval $[-L, L]$, we can write $f(x)$ as the infinite sum of these sines and cosines:

$$f(x) = \frac{a_0}{2} + \sum_{k=1}^{\infty}(a_n \cos(\frac{\pi kx}{L}) + b_n \sin(\frac{\pi kx}{L})), x \in [-L, L]$$

where the coefficients take the form:

$$a_k = \frac{1}{L}\int_{-L}^{L} f(x)\cos(\frac{\pi kx}{L})dx$$
$$b_k = \frac{1}{L}\int_{-L}^{L} f(x)\sin(\frac{\pi kx}{L})dx$$
$$a_0 = \frac{1}{2L}\int_{-L}^{L} f(x)dx$$

2.3 Discrete Fourier Transform

The discrete Fourier Algorithm (DFT) is like a Fourier series but truncated at some maximum frequency. Suppose we are given a sequence of N values that are a function sampled at equally-space points $\{x_0, x_1, x_2, \ldots, x_{N-1}\}$. The DFT is a sequence of numbers given by:

$$\hat{x}_k = \frac{1}{N}\sum_{n=0}^{N-1} x_n e^{\frac{2\pi ikn}{N}}$$

Note that the total complexity of DFT is $\mathcal{O}(N^2)$. The Fast Fourier Transform (FFT), however, is an algorithm that does DFT faster. The complexity of FFT is $\mathcal{O}(Nlog(N))$, and it significantly improves the run time for very large N. We will use the built-in function in MatLab to perform the FFT.

2.4 Averaging

Assuming that white noise is a normally distributed random variable with zero mean to each Fourier component, we can derive something close to the true signal by taking the average over many realizations in the frequency space. Ideally, noise will cancel each other, leaving the non-noise component unaffected. At the same time, we will determine the center frequency as the maximum average frequency. It is reasonable to assume that the true frequency is somewhere around the maximum average frequency as noises are canceling each other.

2.5 Spectral Filter

In order to better detect the signal, we will try to filter out the noise using a spectral filter. In this project, We will use the Gaussian filter around the center frequency and remove the undesired white noise. The Gaussian filter is defined by:

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

Where τ determines the width of the filter and the constant k_0 determines the centre of the filter.
Since our data is obtained from three dimensions, we can adjust this filter using the formula that calculates distance in three dimensions:

$$F(k) = e^{-\tau((k_x-k_{x0})^2+(k_y-k_{y0})^2+(k_z-k_{z0})^2)}$$

Mathematically, we will multiply the signal by the filter in the frequency domain. It is worth mentioning that the Gaussian filter will decay rapidly as it goes away from the center frequency.

3 Algorithm Implementation and Development

Since our goal is to identify the path of the submarine, it is essential to denoise the data. To achieve this, we will divide the code into four main parts:

1. Load data and identify the spatial/frequency domains.
2. Take the average in the frequency domain and locate the center frequency.
3. Apply the Gaussian filter to clean and denoise data.
4. Identify the location and the path of the submarine

Note that the code implemented in MatLab is included in Appendix B.

Algorithm 1: Load Data and identify the spatial/frequency domains

Clean work space
Load data from *subdata.mat*
Set the spatial domain $L = 10$, and the Fourier modes $n = 64$
Create a discrete space x from $-L$ to L with n points starting at $-L$, set $y = x$ and $z = x$
Set the frequency domain k on a 2π periodic signals with $k = \frac{2\pi}{2L}[0 : (\frac{n}{2} - 1) \quad -\frac{n}{2} : -1]$
Set ks as the fftshift of k
Create the 3-D spatial Domain in vector $[X, Y, Z]$, and the frequency domain in $[Kx, Ky, Kz]$

With everything set up, we are now good to start to implement step 2. Note that the size of subdata.mat is $64 \times 64 \times 64 \times 49$, and the last number indicates the time of the data being recorded.

Algorithm 2: Take the average in the frequency domain and locate the center frequency

Initialize avg to be an empty vector
for $t = 1 : 49$ **do**
    Reshape the data on page t in subdata.mat
    Add that data to avg
end
Divide the absolute value of avg by t and store it in avg
Find the location of the maximum in avg
Find the corresponding frequency in the frequency domain.

The frequency we found in the frequency domain is the center frequency we will use in the Gaussian Filter, as mentioned earlier in the Section 2.5. Note that we will take the fast Fourier transform in the next step, and MatLab requires fftshift to obtain the correct transformation.

Algorithm 3: Apply the Gaussian filter to clean and denoise data.

Set $\tau = 0.2$
Initialize the Gaussian filter with the center frequency
for $t = 1 : 49$ **do**
    Reshape the data on page t in subdata.mat
    Take the Fourier transform of that data.
    Multiply that data by the filter.
    Take the Inverse Fourier transform of that data
    Find the location of the max point.
    Find and record the corresponding location in the spatial domain.
end

The location we found in the spatial domain identifies the path of the submarine.

Algorithm 4: Identify the location and the path of the submarine.

For each recorded (x, y, z) at t, plot the data
Output the grid for each (x, y, z).

The results of the computed data (along with the graph) are included in Section 5.

4 Computational Results

Before we filter the subdata, we can plot it to see how the data looks like:

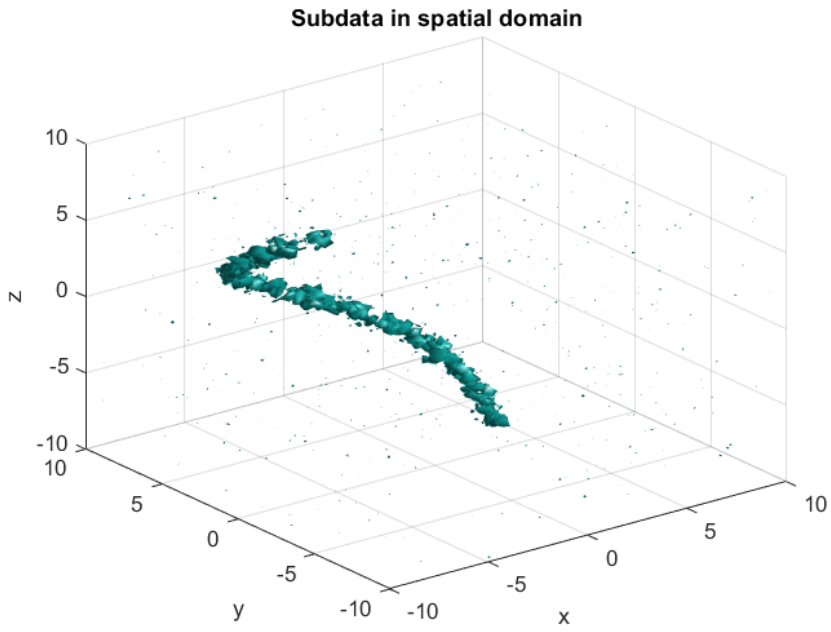

Figure 1: The subdata in spatial domain with a 0.7 threshold

Although we can roughly identify the path of the submarine, it is impossible to determine the submarine's accurate position. Hence, we will use a filter. But first, we need to compute center frequency. For reference, the center frequency we found in the frequency space is $(5.3407, -6.9115, 2.1991)$. We can also find this point in the frequency domain by divide the average by its maximum:

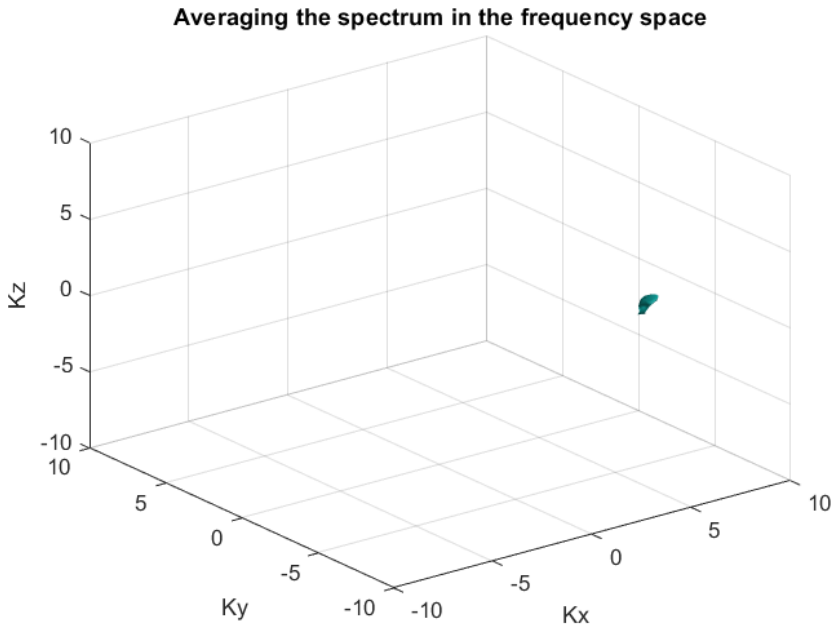

Figure 2: The center frequency in the frequency space with a 0.7 threshold

We can now use the center frequency in the Gaussian filter to clean our data. The submarine's exact location and the plotted path are included in the next section.

5 Summary and Conclusions

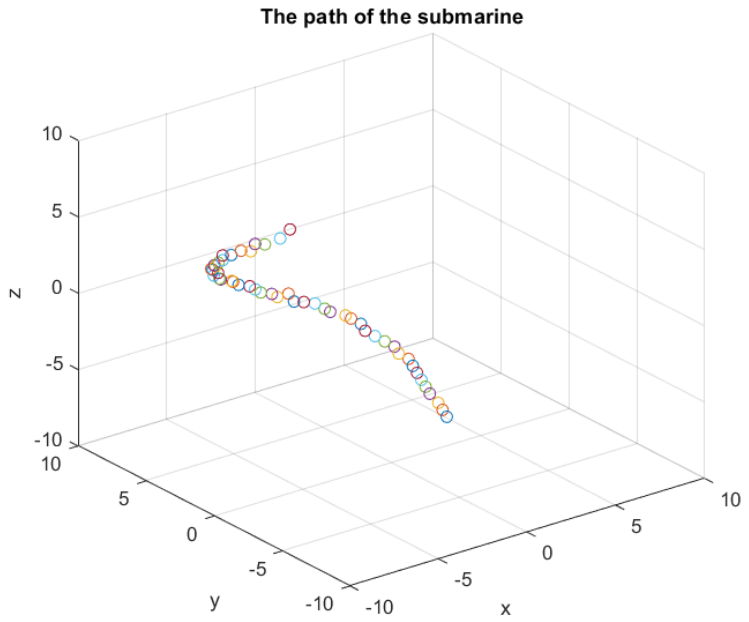

Figure 3: The path of the submarine in spatial domain over the 24-hour period

We can now identify the submarine's location after applying the filter in the frequency space. The Gaussian filter is capable of removing the white noise contained in the data efficiently. Although we are required to use FFT to transform data to the Frequency Space, the reduction of noise in the data set can help us achieve something meaningful. More research can be done on designing a unique filter for various data sets.

time	x	y	z
1	3.1250	0	-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
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	-3.4375
17	1.2500	5	-3.1250
18	0.6250	5.3125	-2.8125
19	0.3125	5.3125	-2.5000
20	0	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

time	x	y	z
26	-2.8125	5.9375	-0.6250
27	-3.1250	5.9375	-0.3125
28	-3.4375	5.9375	0
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	2.1875
36	-6.2500	5	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
45	-6.2500	2.1875	5
46	-6.2500	1.8750	5.6250
47	-5.9375	1.5625	5.6250
48	-5.3125	1.2500	5.9375
49	-5	0.9375	6.5625

Table 1: The location of the submarine

Appendices

Appendix A: MatLab functions used and brief explanations

The following list includes the MatLab functions used in Appendix B. It also contains a brief implementation explanation of their functionalities. More information can be found at MatLab Documentation at MathWorks.

- *y = linspace(x1,x2,n)*: generates n points. The spacing between the points is $\frac{x_2-x_1}{n-1}$
- *y = fftshift(X)*: rearrange a Fourier transform X by shifting the zero-frequency components to the center of the array.
- *[X,Y,Z] = meshgrid(x,y,z)*: returns 3-D grid coordinates defined by the vectors x, y, z.
- *B = reshape(A,sz)*: shapes A using the size vector, sz, to define size(B).
- *M = max(A,[],'all')*: finds the maximum over all elements of A.
- *fv = isosurface(X,Y,Z,V,isovalue)*: computes isosurface data from the volume data V at the isosurface value specified in isovalue.
- *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.
- *X = ifftn(Y)*: returns the multidimensional discrete inverse Fourier transform of an N-D array using a fast Fourier transform algorithm.
- *k = find(X)*: returns a vector with the same orientation as X.
- *[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.
- *Y = exp(X)*: returns the exponential e^x for each element in array X.
- *plot3(X,Y,Z)*: plots coordinates in 3-D space.
- *T = table($var_1, ..., var_N$)*: creates a table from the input variables $var_1, ..., var_N$.
- *writetable(T,filename)*: writes to a file with the name and extension specified by filename.

Appendix B: MatLab Code

```
%% Clean workspace and initialization
clear variables; 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]; ks = fftshift(k);

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

% plot unfiltered subdata
for j=1:49
    Un(:,:,:)=reshape(subdata(:,j),n,n,n);
    M = max(abs(Un),[],'all');
    density = abs(Un)/M;

    isosurface(X,Y,Z,density,0.7);
    axis([-10 10 -10 10 -10 10]),
    xlabel('x'), ylabel('y'),zlabel('z'),
    title('Subdata in spatial domain');
    hold on, grid on, drawnow
    pause(0.2)
end

%% averaging the signal
avg = zeros(n,n,n);
for j = 1:49
    Un(:,:,:)=reshape(subdata(:,j),n,n,n);
    avg = avg + fftn(Un);
end
avg = abs(fftshift(avg))/j;
M = max(avg,[],'all');

% plot the average graph
close all, isosurface(Kx,Ky,Kz, avg/M,0.7)
axis([-10 10 -10 10 -10 10]), grid on, drawnow
title('Averaging the spectrum in the frequency space')
xlabel('Kx'),ylabel('Ky'),zlabel('Kz')

% locate the center frequency
pos = find(avg == M);              % the peak
[r,c,h] = ind2sub(size(avg), pos);
xfreq = Kx(r,c,h);
yfreq = Ky(r,c,h);
zfreq = Kz(r,c,h);

%% define the Gaussian filter
tau = 0.2;
filter = exp(-tau*((Kx-xfreq).^2 + (Ky-yfreq).^2 + (Kz-zfreq).^2));

% find the submarine
xpos = zeros(49,1); ypos = xpos; zpos = xpos;
for j = 1:49
    Un(:,:,:) = reshape(subdata(:,j),n,n,n);
```

```
58      Untf = fftshift(fftn(Un)).* filter;
59      Unf = ifftn(fftshift(Untf));
60      M = max(Unf, [], 'all');
61
62      pos = find(Unf == M);
63      [r, c, h] = ind2sub(size(Unf), pos);
64      xpos(j) = X(r,c,h); ypos(j) = Y(r,c,h); zpos(j) = Z(r,c,h);
65
66      % plot the data
67      plot3(xpos(j),ypos(j),zpos(j), '-o');
68      xlabel('x'), ylabel('y'),zlabel('z'),
69      xlim([-10,10]),ylim([-10,10]),zlim([-10,10])
70      % title(['(',num2str(xpos),', ',num2str(ypos),', ',num2str(zpos),') when t =
            ', num2str(j)]);
71      title('The path of the submarine');
72      hold on, grid on, drawnow
73      pause(0.2)
74  end
75
76  %% output the chart
77  clc;
78  time = (1:49)';
79  t = table(time,xpos, ypos, zpos);
80  writetable(t,'data.csv');
```