A Submarine Problem

Luotong Kang

Abstract

A moving submarine in the Puget Sound is emitting an unknown acoustic frequency. Through the use of the Discrete Fourier Transform, acoustic signature and filtering by frequency, we can locate the submarine and find its trajectory.

1 Introduction

Using a broad spectrum recoding of acoustics, data is obtained over a 24-hour period in half-hour increments. Unfortunately, the submarine is moving in a unknown path. To determined its path and location, we need to first average 49 realizations to get the frequency signature generated by the submarine. Then, we can determine the path of submarine over the past 24 hours by denoising the 3-dimensional data. By looking through the table of x,y coordinates, we are able to send P-8 Poseidon subtracking aircraft to the final location of submarine.

2 Theoretical Background

An integral transform of the $f(x)$ is a relation of the form:

$$F(s) = \int_{\alpha}^{\beta} K(s,t) f(t) dt$$

where $K(s,t)$, α and β are given.The function $K(s,t)$ is called the kernel of the transformation and α, β could be $\pm\infty$. This relation transforms the function $f(t)$, to another function $F(s)$, called the transform of $f(t)$.

Applying appropriate transform to raw data to make them interpretable is very important in data-analysis. The transform we are going to use here is called *The Fourier Transform*, which allows us to represent a given function as sines and cosines of different frequency.

Suppose we are given a function $f(x)$ with $x \in \mathbb{R}$. We define the *The Fourier Transform* of $f(x)$, written $\hat{f}(x)$, using the formula

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

Furthermore, the *Inverse Fourier Transform* is defined as:

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

Using the *Euler's formula*, the frequency k can be easily understood and plotted.

However, we need a discrete version of this transform (DFT) in order to analyze the data in finite domain:

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

Since our purpose of doing DFT in submarine problem is to denoise the data, we're going to multiply the signal filter with a filter in frequency domain around the center frequency. As long as one function is localized and goes to zero exponentially, it can definitely be a filter. Here in convenience we just choose to use the Gaussian function:

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

where τ is the width of the filter and k_0 determining the centre of the filter.

3 Algorithm Implementation and Development

Using MATLAB, we first plot the raw data with small isovalue to see the general path for every time step and them putting all 49 realizations in one graph:

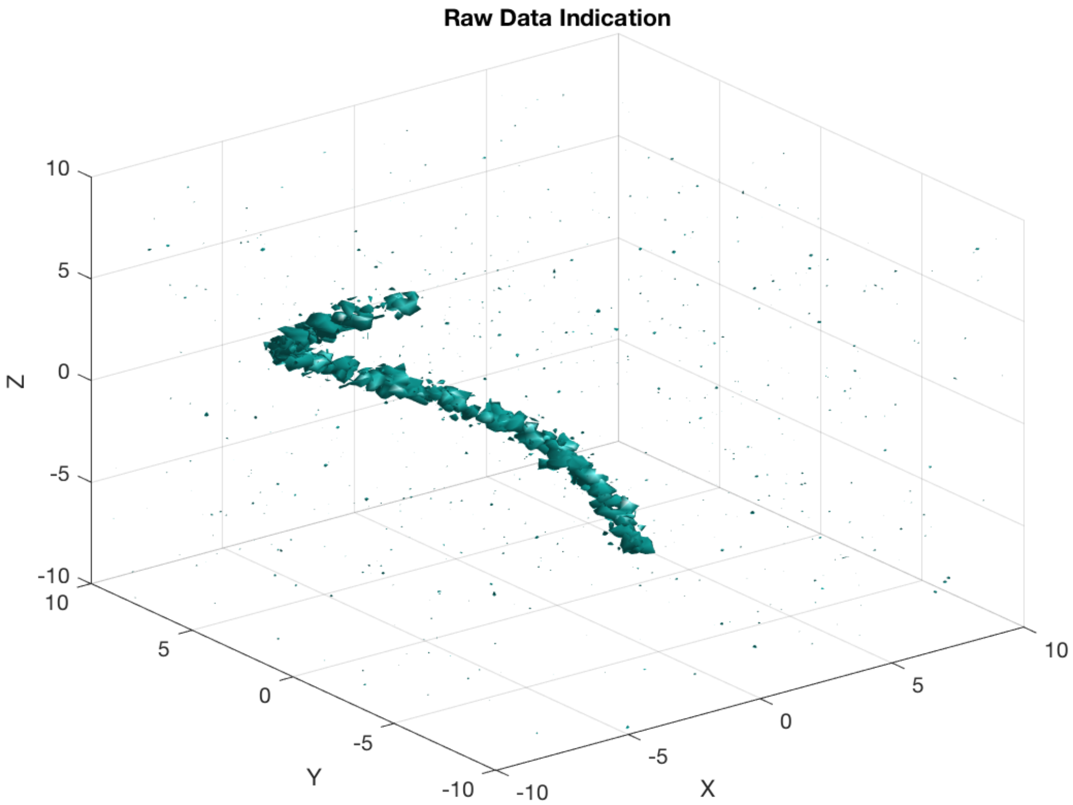

Figure 1: Raw Data Indication.

It is tricky to implement DFT because of the aliasing: instead of thinking the result is in the sequence of $k = 0, 1, 2, ..., N-1$, we should consider the frequencies are given by $k = -N/2, N/2+1, ..., -1, 0, 1, ..., N/2-1$. In addition, MATLAB command `fft/fftn` give us results in the order $\hat{x_0}, \hat{x_0}, ..., x_{-\hat{N/2-1}}, x_{-\hat{N/2}}, x_{-\hat{N/2+1}}, ..., \hat{x_{-1}}$. Thus we have to use the command `fftshift` to let the result make sense. In addition to rescaling the frequencies by $2\pi/L$ (`fftn` assumes 2π periodic signals), we also want to do `fftshift` beforehand to avoid doing it all the time when plotting:

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

After solving the MATLAB implementation of DFT, we are able to find the center frequency: We first transform each realization in frequency domain (make sure to use `fftshift` to obataine the correct order) and add them together to get a 64x64x64 matrix. Then divide it by 49 (number of realization) element-wise to get the averaged frequency data. Use `M=max(abs(Unave),[],'all')` to find the highest frequency (the signature) and use `find(abs(Unave)==M)` to get the indices of the center frequency. Lastly, call out the corresponding element in `Kx,Ky,Kz`.

Using the center frequency founded, we implement the 3-dimensional version of Gaussian function by coding

```
filter = exp(-tau.*((Kx-Kx0).^2 + (Ky-Ky0).^2 + (Kz-Kz0).\^2)).
```

In each realization, we multiply the filter with the data in frequency domain and find the indices of largest frequency. After getting 49 sets of indices, we can eventually plot the path of submarine:

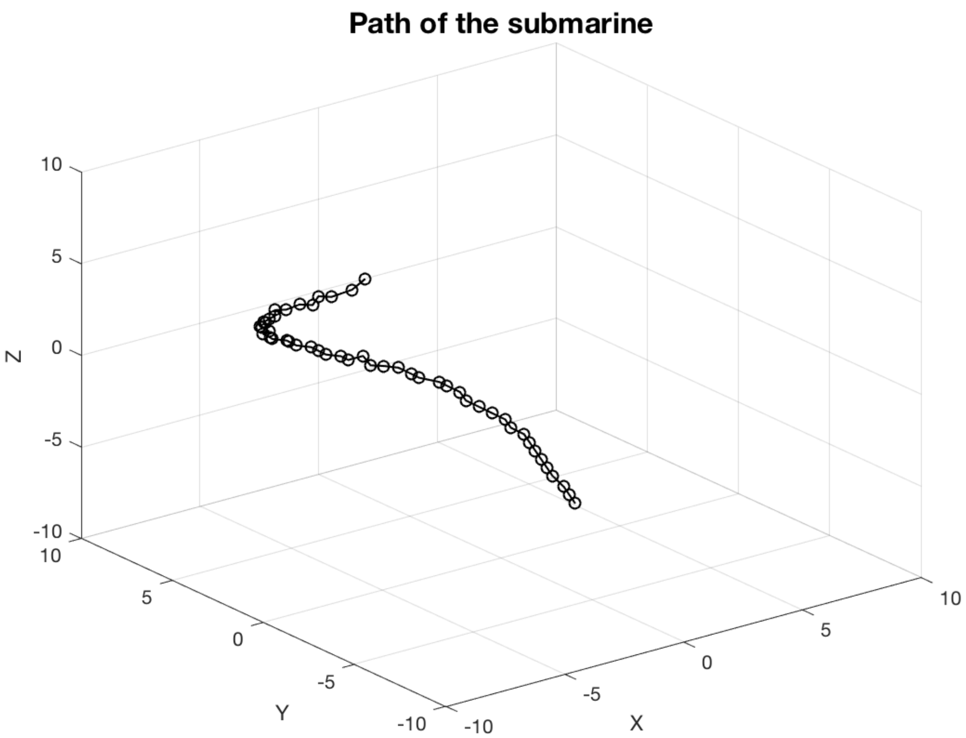

Figure 2: Path of the submarine.

4 Computational Results

The center frequency is $[Kx0, Ky0, Kz0] = [5.3407, -6.9115, 2.1991]$. The location of submarine in each realization is given in the table below:

	x	y		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	2.5	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: Location of submarine in all 49 realizations.

5 Summary and Conclusions

In the process of finding a mysterious submarine, we apply spectrum averaging, filter denoising, and 3D plotting using MATLAB and successfully track the submarine. From the computational result, we can conclude that the lastest location of the submarine we've detected is $(x, y) = (-5, 0.9375)$. We should send our P-8 Poseidon subtracking aircraft to that location in the Puget Sound.

6 Appendices

6.1 MATLAB functions used

The commands are listed chronologically.
`load`: imports a variable into the Workspace.
`linspace(a,b,n)`: creates a vector with n elements linearly spaced between a and b
`meshgrid(x,y,z)`: returns the 3D grid coordinates defined by the vectors x, y, z
`reshape(A,[sz])`: reshapes A into a matrix of the size specified by [sz]
`max(A,[],'all')`: returns the largest element in A `abs(A)`: takes the absolute value of every element in A, also works for complex numbers
`isosurface(X,Y,Z,V,iv)`: returns an isosurface plot of 3D data sets V which connects points with the specified isovalue=iv, and has axes specified by X,Y, and Z,
`print('a','-dpng')`: saves the figure we just plotted as a png file named a.
`fftshift(A)`: shifts the zero-frequency component to the center of spectrum
`fftn(A)`: returns the N-dimensional discrete Fourier transform of A
`find(A==n)`: returns the linear indices of the element n in A
`[I1,I2,I3]=ind2sub([sz],ind)`: returns the size=[sz] indices equivalent to the linear index ind
`exp(n)`: the exponential of the elements of n, e to the n
`ifftn(A)`: returns the N-dimensional inverse discrete Fourier transform of A
`plot3(X,Y,Z)`: plots coordinates in 3D space

6.2 MATLAB codes

```matlab
clear all; close all; clc;

load subdata.mat

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);

for j=1:49
    Un(:,:,:)=reshape(subdata(:,j),n,n,n);
    M = max(abs(Un),[],'all');
    isosurface(X,Y,Z,abs(Un)/M,0.7)
    title('Raw Data Indication')
    xlabel('X')
    ylabel('Y')
    zlabel('Z')
    axis([-10 10 -10 10 -10 10]), grid on, hold on
end
print('Raw Data Indication','-dpng')

```

```
%% Center Frequency
Unave=zeros(n,n,n);
for j=1:49
    Unave = Unave+fftshift(fftn(reshape(subdata(:,j),n,n,n)));
end
Unave = Unave./49;
M = max(abs(Unave),[],'all');
indices2=find(abs(Unave)==M);
[kx0,ky0,kz0] = ind2sub([n,n,n],indices2);
Kx0=Kx(kx0,ky0,kz0);
Ky0=Ky(kx0,ky0,kz0);
Kz0=Kz(kx0,ky0,kz0);

%% Filter and Plot
tau = 0.2;
filter = exp(-tau.*((Kx-Kx0).^2 + (Ky-Ky0).^2 + (Kz-Kz0).^2));
for j=1:49
    utn = fftshift(fftn(reshape(subdata(:,j),n,n,n)));
    uft = filter.* utn;
    ufti = ifftn(uft);
    M = max(abs(ufti),[],'all');
    indices2=find(abs(ufti)==M);
    [mx,my,mz] = ind2sub([n,n,n],indices2);
    subX(j) = X(mx,my,mz);
    subY(j) = Y(mx,my,mz);
    subZ(j) = Z(mx,my,mz);
end
sub = [subX;subY;subZ].';
plot3(subX,subY,subZ, 'ko-','Linewidth',1)
title('Path of the submarine','Fontsize',15)
xlabel('X')
ylabel('Y')
zlabel('Z')
axis([-10 10 -10 10 -10 10]), grid on
print('Path of the submarine','-dpng')
```