AMATH 482 Homework 1

Tsz Wai Tsui

January 27, 2021

Abstract

We aim to locate and find the trajectory of a moving submarine in the Puget Sound by using noisy acoustic data emitted by it. By averaging the spectrum to find the center frequency and applying a filter on the data around the center frequency to denoise it, we can determine the path of the submarine so that we know where to send our P-8 Poseidon subtracking aircraft.

1 Introduction and Overview

We are given a 262144x49 (space by time) noisy acoustic data obtained over a 24-hour period in half-hour increments. Since the moving submarine is using new technology, the frequency we need to detect is unknown. To determine the frequency signature generated by the submarine, we need to first apply Fourier Transform on the data to transform the data from the spatial domain to the frequency domain and then average the data. After that, we apply the Gaussian function as a filter on the data around the center frequency to denoise data and determine the path of the submarine. The 49th data point is where we should send our P-8 Poseidon subtracking aircraft to.

2 Theoretical background

Fourier Transform takes a function of space or time and converts it to a function to frequencies by the formula:

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

The inverse of Fourier Transform takes a function of frequencies and converts it back to a function of space of time by the formula:

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

Fourier Series is a way of expressing a periodic function as the infinite sum of sines and cosines. Given a function $f(x)$ for a limited range of x values and for k = 1, 2, 3, 4, … , it can be written as:

$$f(x) = \frac{a_0}{2} + \sum_{k=1}^{\infty}(a_n\cos(kx) + b_n\sin(\mathrm{kx}))\,, x \in [-\pi, \pi]. \tag{3}$$

In this case, we use Discrete Fourier Transform because we are given a discrete set of data points instead of a function. For N values sampled at equally-spaced $\{x_0, x_1, x_2, \dots, x_{N-1}\}$, the formula is:

$$\hat{x}_k = \frac{1}{N}\sum_{n=0}^{N-1} x_n e^{\frac{2\pi ikn}{N}}. \tag{4}$$

In addition, we use the Fast Fourier Transform (FFT), which computes the Discrete Fourier Transform faster. The total complexity of FFT is O(Nlog(N)), while the one of DFT is O(N^2). Therefore, FFT takes a lot fewer operations when N is large.

To denoise the data, we average and apply a filter to it. Averaging the spectrum is an effective way to find out the center frequency of the submarine because white noise affects all frequencies the same, having a zero mean. When we average over many realizations in the frequency domain, the white noise is added up to zero. As a result, we can get the center frequency and proceed to the next step: filtering the data.

We use the Gaussian function to filter the data around the center frequency to remove undesired frequencies. In the formula below, τ is the width of the filter and the constant k_0 is the center of the filter.

$$F(k) = e^{-\tau(k-k_0)^2}. \tag{5}$$

Below is the Gaussian function we use in 3-D.

$$F(k) = e^{-\tau((k_x-k_{x0})^2+(k_y-k_{y0})^2+(k_z-k_{z0})^2)}. \tag{6}$$

3 Algorithm Implementation and development

First, we load `subdata.mat` into MATLAB and define the spatial domain, Fourier modes, and grid vectors. It is important to note that we rescale the frequencies by $\frac{2\pi}{L}$ since FFT assumes 2π periodic signals. We use the `reshape` command to transform the data from 262144x49 to 64x64x64. Then, we use `fftn` command to apply FFT on the data, converting it from spatial domain to frequency domain in a for loop. However, since the data is noisy, we sum up the data, use `fftshift` command to shift the data, and divide its absolute value by 49 to average out white noise. After that, we use `max` and `ind2sub` commands to find out the maximum value of the averaged data and get its indices. By plugging the indices into `Kx`, `Ky` and `Kz`, we get the x, y, and z coordinates of the center frequency.

Since we know where to apply the Gaussian filter around, we plug in the coordinates of center frequency into the Gaussian function and multiply it by the data in the frequency domain. To do this, we set up a for loop to again `reshape` the data and apply FFT using `fftn`. Different from averaging the spectrum, we multiply the data by the filter and transform it back to the spatial domain using `ifftn` command. Then, we find out the indices of the maximum signal and plug them into `X`, `Y`, and `Z`. In the loop, we record their x, y, and z coordinates in separate vectors to visualize the path of the submarine at the end using the `plot3` command.

4 Computational Results

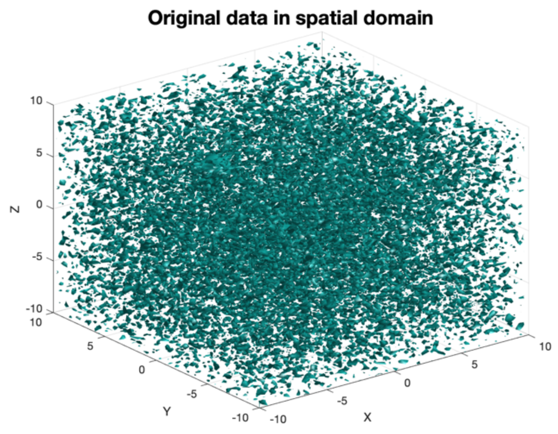

Figure 1: It shows the noisy original data in an isosurface plot before FFT

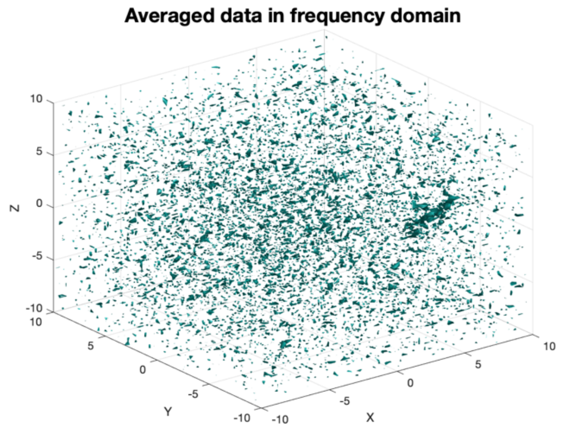

Figure 2: It shows the averaged FFT transformed data in an isosurface plot

After averaging the spectrum, we found that the center frequency locates at (5.3407, -6.9115, 2.1991), where is the condensed area in Figure 2.

As we know where to filter around, we can plug the coordinates into the 3-D Gaussian function to denoise the data. Through doing inverse FFT to convert the data back to the spatial domain, we obtain the x, y, and z coordinates of the submarine's location in each of the 49 data points. Figure 3 shows the path of submarine, while Table 1 - 6 show the corresponding x and y coordinates. At the 49th point, the submarine is located at (-5, 0.9375, 6.5625).

x	3.125	3.125	3.125	3.125	3.125	3.125	3.125	3.125	3.125
y	0	0.3125	0.625	1.25	1.5625	1.875	2.1875	2.5	2.8125

Table 1: x and y coordinates of submarine from data points 1 to 9

x	2.8125	2.8125	2.5	2.1875	1.875	1.875	1.5625	1.25	0.625
y	3.125	3.4375	3.75	4.0625	4.375	4.6875	5	5	5.3125

Table 2: x and y coordinates of submarine from data points 10 to 18

x	0.3125	0	-0.625	-0.9375	-1.25	-1.875	-2.1875	-2.8125	-3.125
y	5.3125	5.625	5.625	5.9375	5.9375	5.9375	5.9375	5.9375	5.9375

Table 3: x and y coordinates of submarine from data points 19 to 27

x	-3.4375	-4.0625	-4.375	-4.6875	-5.3125	-5.625	-5.9375	-5.9375	-6.25
y	5.9375	5.9375	5.9375	5.625	5.625	5.3125	5.3125	5	5

Table 4: x and y coordinates of submarine from data points 28 to 36

x	-6.5625	-6.5625	-6.875	-6.875	-6.875	-6.875	-6.875	-6.5625	-6.25
y	4.6875	4.375	4.0625	3.75	3.4375	3.4375	2.8125	2.5	2.1875

Table 5: x and y coordinates of submarine from data points 37 to 45

x	-6.25	-5.9375	-5.3125	-5
y	1.875	1.5625	1.25	0.9375

Table 6: x and y coordinates of submarine from data points 46 to 49

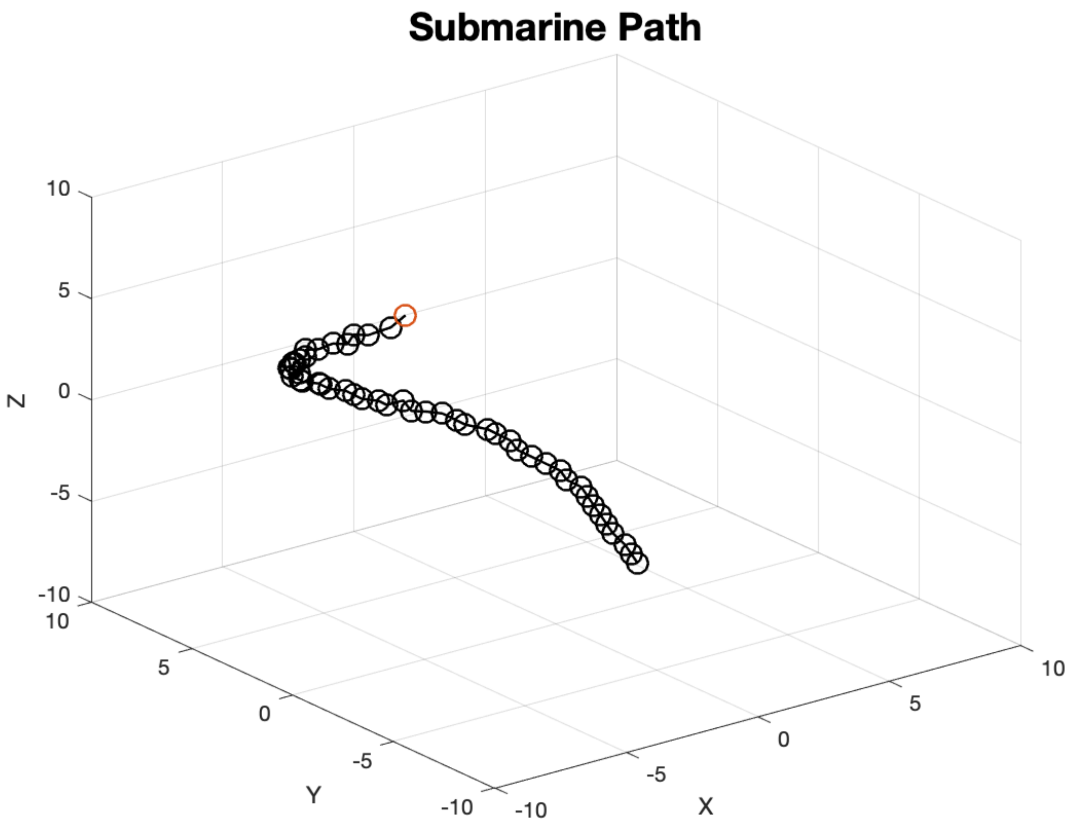

Figure 3: It shows the path of submarine over time in 3-D. The red circle marks its final location.

5 Summary and Conclusions

In conclusion, through using Fast Fourier Transform, we are able to transform reshaped data in the spatial domain to the frequency domain. To locate the submarine, we need to denoise the data and find out the center frequency by averaging the spectrum. After that, we apply the Gaussian filter on it and get the x, y, and z coordinates of submarine at all 49 data points. As shown in Figure 3, we are able to locate the final position of the submarine and visualize its trajectory. We now can send our P-8 Poseidon subtracking aircraft to (-5, 0.9375, 6.5625).

Appendix A. MATLAB functions

- `load filename` loads data from `filename`
- `y = linspace(x1, x2, n)` returns a row vector of `n` evenly spaced points between `x1` and `x2`.
- `[X, Y, Z] = meshgrid(x, y, z)` returns 3-D grid coordinates based on the coordinates contained in the vectors `x`, `y`, and `z`. The grid represented by the coordinates `X`, `Y` , and `Z` has size `length(y)`-by-`length(x)`-by-`length(z)`.
- `B = reshape(A,sz)` reshapes `A` using the size vector, `sz`, to define `size(B)`. For example, `reshape(A,[2,3])` reshapes `A` into a 2-by-3 matrix. `sz` must contain at least 2 elements, and `prod(sz)` must be the same as `numel(A)`.
- `Y = fftn(X)` returns the multidimensional Fourier transform of an N-D array using a fast Fourier transform algorithm. The N-D transform is equivalent to computing the 1-D transform along each dimension of `X`. The output `Y` is the same size as `X`.
- `Y = fftshift(X)` rearranges a Fourier transform `X` by shifting the zero-frequency component to the center of the array. If `X` is a vector, then `fftshift` swaps the left and right halves of `X`. If `X` is a matrix, then `fftshift` swaps the first quadrant of `X` with the third, and the second quadrant with the fourth. If `X` is a multidimensional array, then `fftshift` swaps half-spaces of `X` along each dimension.
- `X = ifftn(Y)` returns the multidimensional discrete inverse Fourier transform of an N-D array using a fast Fourier transform algorithm. The N-D inverse transform is equivalent to computing the 1-D inverse transform along each dimension of `Y`. The output `X` is the same size as `Y`.
- `fv = isosurface(X,Y,Z,V,isovalue)` computes isosurface data from the volume data `V` at the isosurface value specified in `isovalue`. That is, the isosurface connects points that have the specified value much the way contour lines connect points of equal elevation.
- `[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`. Here `sz` is a vector with two elements, where `sz(1)` specifies the number of rows and `sz(2)` specifies the number of columns.
- `X = zeros(sz1,...,szN)` returns an `sz1`-by-...-by-`szN` array of zeros where `sz1`, ... , `szN` indicate the size of each dimension. For example, `zeros(2,3)` returns a 2-by-3 matrix.

- `M = max(A)` returns the maximum elements of an array. If `A` is a vector, then `max(A)` returns the maximum of `A`. If `A` is a matrix, then `max(A)` is a row vector containing the maximum value of each column. If `A` is a multidimensional array, then `max(A)` operates along the first array dimension whose size does not equal 1, treating the elements as vectors. The size of this dimension becomes 1 while the sizes of all other dimensions remain the same. If `A` is an empty array whose first dimension has zero length, then `max(A)` returns an empty array with the same size as `A`.
- `Y = abs(X)` returns the absolute value of each element in array `X`. If `X` is complex, `abs(X)` returns the complex magnitude.
- `plot3(X,Y,Z)` plots coordinates in 3-D space. To plot a set of coordinates connected by line segments, specify `X`, `Y`, and `Z` as vectors of the same length. To plot multiple sets of coordinates on the same set of axes, specify at least one of `X`, `Y`, or `Z` as a matrix and the others as vectors.

Appendix B. MATLAB codes

```
% Clean workspace
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];
ks = fftshift(k);

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

Utavg = zeros(n,n,n);
for j=1:49
    Un(:,:,:)=reshape(subdata(:,j),n,n,n);
    Utavg = Utavg + fftn(Un);
    M = max(abs(Un),[],'all');
    close all, isosurface(X,Y,Z,abs(Un)/M,0.7)
    axis([-20 20 -20 20 -20 20]), grid on, drawnow
    pause(1)
end
% graph of untransformed data
isosurface(X,Y,Z,abs(Un),0.4)
axis([-10 10 -10 10 -10 10]),grid on
title('Original data in spatial domain', 'Fontsize', 18)
xlabel('X')
ylabel('Y')
zlabel('Z')
print('Figure 1','-dpng');
```

```
close all;
% graph after averaging
Utavg = abs(fftshift(Utavg))./49;
isosurface(Kx,Ky,Kz,Utavg./max(Utavg),0.85)
axis([-10 10 -10 10 -10 10]),grid on
title('Averaged data in frequency domain', 'Fontsize', 18)
xlabel('X')
ylabel('Y')
zlabel('Z')
print('Figure 2','-dpng');

close all;
[maxi, indx] = max(abs(Utavg(:)));
[x_indx, y_indx, z_indx] = ind2sub([n,n,n], indx);
x_center = Kx(x_indx, y_indx, z_indx);
y_center = Ky(x_indx, y_indx, z_indx);
z_center = Kz(x_indx, y_indx, z_indx);

tau = 0.2;
filter = fftshift(exp(-tau.*((Kx-x_center).^2+ ...
(Ky-y_center).^2+(Kz-z_center).^2)));

x_arr = zeros(49,1);
y_arr = zeros(49,1);
z_arr = zeros(49,1);

for i = 1:49
    Utn = fftn(reshape(subdata(:,i),n,n,n));
    Unft = filter.*Utn;
    Unf = ifftn(Unft);
    [f_maxi, f_indx] = max(abs(Unf(:)));
    [xf_indx, yf_indx, zf_indx] = ind2sub([n,n,n], f_indx);
    x_arr(i) = X(xf_indx, yf_indx, zf_indx);
    y_arr(i) = Y(xf_indx, yf_indx, zf_indx);
    z_arr(i) = Z(xf_indx, yf_indx, zf_indx);
end

plot3(x_arr,y_arr,z_arr,'k-o','LineWidth', 1.2,'MarkerSize',10),
axis([-10 10 -10 10 -10 10]), grid on
hold on
plot3(x_arr(49),y_arr(49),z_arr(49),'-o','LineWidth', 1.2,'MarkerSize',10)
title('Submarine Path','Fontsize', 18)
xlabel('X')
ylabel('Y')
zlabel('Z')
print('Figure 3','-dpng');
```