AMATH 482 Homework 1

Jessica Shi, 1762333

January 14, 2021

Abstract

In this homework, we use proper data averaging and filter techniques to denoise the given spectrum data and locate the submarine. The key idea is to use the Fourier Transform to determine the center frequency and the Gaussian function to filter out the noise. Then we are able to keep track of the position of the submarine in each time step. And finally we visualize the path of the submarine.

1 Introduction and Overview

Suppose you are searching for a submarine. The only data given was an acoustic data obtained in a 24-hour period. The data was recorded per half hour. You need to detect the acoustic frequency generated by the new submarine technology, find the location of the submarine and determine its path.

The whole path of submarine is under water and the noise might be the result of fluid's and other creatures' movements. Thus, the noise would be in some random frequencies. Then it is reasonable to assume that if we average the spectrum data, the random frequencies would cancel out each others. Consequently, we could get the center frequency produced by the submarine itself. Then, we could use a filter to further clean the data, get the location of the submarine at each time step and find the path of the submarine in this time period.

2 Theoretical Background

In order to finish our task of locating and finding the path of submarine, we need to understand 3 key ideas: Averaging the spectrum data, the Fourier Transform and the Gaussian Filtering. These concepts help us to find the center frequency and denoise the given data.

2.1 Averaging Data

From the lecture, we know that usually the noise in the nature is the white noise. And the white noise affects all the frequencies the same. More importantly, it has a zero mean, meaning that if we average over all the data, the noise from different realizations would cancel out and we could finally get the data that is closer to the actual signal data. However, this method has a possible flaw. By averaging the data, the frequency domain is clear and we know about the high and low frequencies, but we would lose the information about the position of the frequency in the time domain. Here, our main purpose of averaging the data is to find the central frequency so that we could apply the Gaussian filter to denoise the data.

2.2 The Fourier Transform and Fast Fourier Transform

Given $f(x)$, the function, the Fourier Transform first multiply it with the Euler's Formula and then integrate it over an infinite domain. As a result, we could get the frequency function $F(k)$.

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

The Fourier Transform (FT) decomposes the given time functions into functions in the frequency domain. The result is represented by the sum of some sine and cosine.

Furthermore, if we are given the function $F(k)$ after the transform, we could use the inverse Fourier Transform to get the original function. And it helps us to change the data from the frequency domain back to the time domain.

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

For faster and better computation, we use the Fast Fourier Transform (FFT) algorithm. It is a builtin function in MATLAB and it has a higher accuracy. When using the FFT, we need to first change the domain to $[-2\pi, 2\pi]$ to re-scale the given data. Also, since the function use the "divide and conquer" algorithm, we need to assume the data are in N points, where N should be equal to 2^i and i should be integer.

2.3 Gaussian Filtering

A proper way to denoise the data mentioned in the lecture is to filter out the undesirable data that is not around a certain frequency. We could multiply the spectrum by a function to get only desired data left. There are many possible functions, among which the Gaussian function is simple to use:

$$e^{-\tau(k-k_0)^2} \tag{3}$$

This is the Gaussian function. The constant k_0 determines the center of the filter and the degree of amplification is determined by τ. After multiplying our data by this function, the data around the center frequency would be amplified so that we could filter out unrelated data and keep the desired ones.

3 Algorithm Implementation and Development

The specific implementation is done using MATLAB and is divided into several steps according to the problem requirements.

3.1 Problem Setup

First, I load the data into MATLAB and define the time domain L from -10 to 10 as my spatial domain. I divide all the data along x, y, z axes into 64 points, where 64 is my Fourier modes. Then, I multiply the time domain with $\frac{2\pi}{2L}$ so that I could get the scaled frequency domain from -2π to 2π.

3.2 Spectrum Averaging and Center Frequency Determination

The data has 49 columns which represents 49 time steps of spectrum recording. Thus, I loop over all the columns to do `fft` for every column and add all the resulting values together. Also, it is better to normalized the data to make them in the range from 0 to 1. After the loop is done, I then calculate the average of spectrum data. The maximum value after this averaging process would be most related to the actual signal from the submarine, so I use a function `max` to find the maximum frequency along each axis and its corresponding index. `kx_c, ky_c, kz_c` give us the location of the center frequency.

3.3 Filter Application and Plotting

After finding the center frequency, we could apply the Gaussian function to filter the spectrum data to get rid of unrelated noises. Based on the equation mentioned in the Theoretical Background part, I define the filter function. Similar to previous section, I loop over all 49 columns of data, take the `fft` of data. Also, I use the `fftshift` to make it have the proper order. Then I apply the filter by multiplying the given data with the filter function. During this process, different values of τ have been tried. And I think 0.2 would be a proper choice.

After denoising, I shift the data back using `ifftshift` and get the original submarine's positions in the time

domain using `ifft`. Then, I use the same function to find the location of the maximum frequency, which is the location of the submarine at that time step. After the loop is done, we could get the coordinates of the locations of submarine at all time steps. Finally, we could use these coordinates to plot the path of the submarine.

4 Computational Results

From the second section of implementation, the central frequency is at (5.3407, -6.9115, 2.1991). And the last position of the submarine is at (-5, 0.9375, 6.5625). The complete path of the submarine is shown in Figure 1 below.

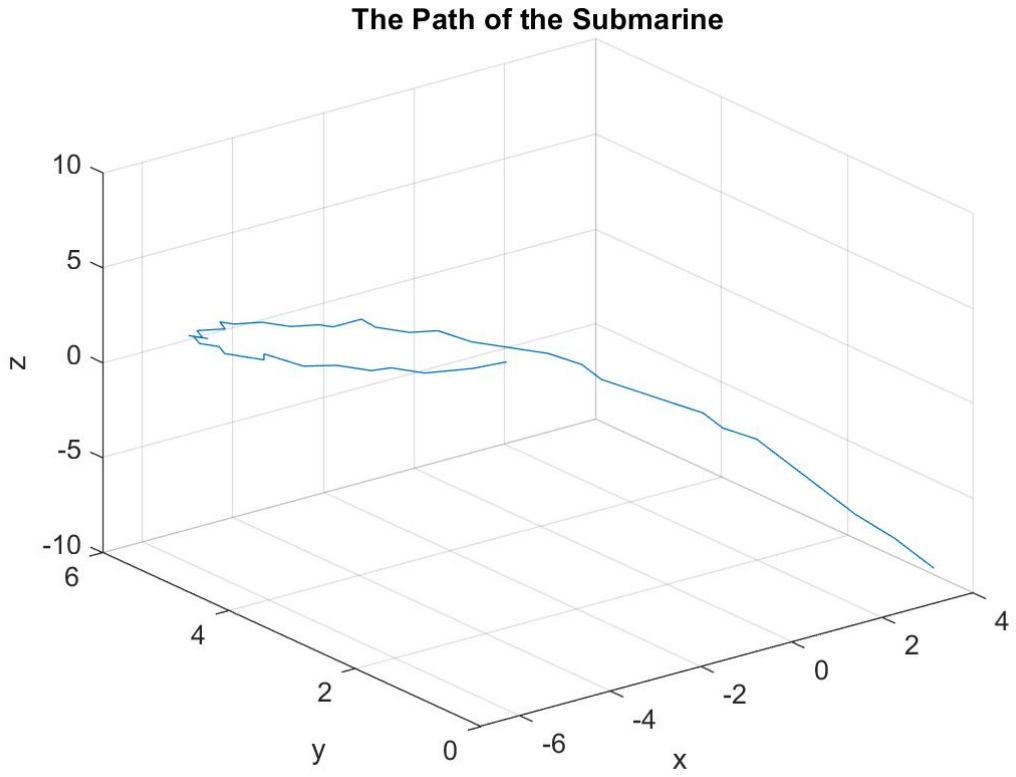

Figure 1: The Path of the Submarine

5 Summary and Conclusions

To conclude, in order to find the center frequency of the given spectrum data, we could use the Fourier Transform and the Fast Fourier Transform function in MATLAB to average the data and then to find the center frequency. The center frequency is at (5.3407, -6.9115, 2.1991). After finding the center frequency, we could apply the Gaussian function as a filter to clean the data, and use the inverse FFT to convert the data back to the time domain. Then we could determine the position of the submarine at each time step and finally get the path of the submarine. In summary, the techniques of averaging data, Fourier Transform and Gaussian function filter help us to find the related, useful data from the noisy given spectrum data.

Appendix A MATLAB Functions

Add your important MATLAB functions here with a brief implementation explanation.

- `y = linspace(x1,x2,n)` returns a row vector of `n` evenly spaced points between `x1` and `x2`.
- `[X,Y] = meshgrid(x,y)` returns 2-D grid coordinates based on the coordinates contained in the vectors `x` and `y`. X is a matrix where each row is a copy of `x`, and `Y` is a matrix where each column is a copy of `y`. The grid represented by the coordinates `X` and `Y` has `length(y)` rows and `length(x)` columns.
- `Y = fftshift(X)` rearranges a Fourier transform `X` by shifting the zero-frequency component to the center of the array.
- `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 = ifftn(X)` 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`.
- `Y = ifftshift(X)` undoes the result of `fftshift`.
- `[M,I] = max(A)` returns the index into the operating dimension that corresponds to the maximum value of A.
- `plot3(X,Y,Z)` plots coordinates in 3-D space.

Appendix B MATLAB Code

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

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

L = 10; % spatial domain
n = 64; % Fourier modes
x2 = linspace(-L, L, n+1);
x = x2(1:n); % only use the first n points
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);


% PART 1: Average the spectrum, determine the center frequency. -----------
u_ave = zeros(n, n, n);

for i = 1:49
    Un(:,:,:)=reshape(subdata(:,i), n, n, n);
    unt = fftn(Un);
    u_ave = u_ave + unt;
end
u_ave = abs(fftshift(u_ave)) / max(max(max(abs(u_ave))));
[M, I] = max(u_ave(:));

% find the center frequency's location
```

```
kx_c = Kx(I);
ky_c = Ky(I);
kz_c = Kz(I);


% for j=1:49
%     Un(:,:,:)=reshape(subdata(:,j),n,n,n);
%     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


% PART 2: Filter/Denoise the data. ---------------------------------------

% Define the filter
tau = 0.2;
filter = exp(-tau*((Kx - kx_c).^2 + (Ky - ky_c).^2 + (Kz - kz_c).^2));

position = zeros(n, 3);

path_x = zeros(49, 1);
path_y = zeros(49, 1);
path_z = zeros(49, 1);


for i = 1:49
    Un(:,:,:)=reshape(subdata(:,i), n, n, n);
    unft = ifftshift(filter.*fftshift((fftn(Un))));
    pos = ifftn(unft);
    pos = pos / max(max(max(pos)));
    [M, I] = max(pos(:));
    path_x(i, 1) = X(I);
    path_y(i, 1) = Y(I);
    path_z(i, 1) = Z(I);

    % help plot the position of submarine at each time step
%     isosurface(X, Y, Z, abs(pos), 0.4)
%     axis([-20 20 -20 20 -20 20]), grid on, drawnow
%     hold on;
end


% Find the last location of the submarine
end_pos = [path_x(49), path_y(49), path_z(49)];

close all;

% Plot the path of the submarine over the whole time period
plot3(path_x, path_y, path_z);
xlabel('x');
ylabel('y');
zlabel('z');
title('The Path of the Submarine');
grid on
```