AMATH 482 Homework 1

Biyao Li

January 21, 2021

Abstract

In this assignment, I will investigate a procedure to determine the location of a submarine by detecting a frequency from a noisy acoustic data. This assignment relies heavily on the use of Fourier transform and several related MATLAB functions.

1 Introduction and Overview

A submarine is active in the Puget Sound and we are trying to get the path of the submarine so that we can send P-8 Poseidon subtracking aircraft to its final location. The submarine possess a new technology that emits a frequency which can be used to find its position and a set of noisy acoustic data was collected every 30 minutes in a 24 hour time period. This data have 262144 rows and 49 columns and each column will be processed into a 64×64×64 3-D matrix.

To find the frequency emitted by the submarine, we need to apply Fourier Transform to data in each of the 49 columns so that the data will be put into the frequency domain. Then we can find the frequency by averaging the Fourier Transforms and apply a filtering function around the target frequency to remove unnecessary frequencies and noises. After data is denoised, we can use inverse Fourier transfer to put data back in the spatial domain which will give us information about the location of the submarine.

2 Theoretical Background

2.1 Fourier Transform

Joseph Fourier introduced the concept of representing a given function $f(x)$ by a trigonometric series of sines and cosines:

$$f(x) = \frac{a_0}{2} + \sum_{i=1}^{\infty} (a_n \cos nx + b_n \sin nx) \quad x \in (-\pi, \pi]. \tag{1}$$

Moreover, given any function $f(x)$, the Fourier transform of $f(x)$ is the following:

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

And to convert $\hat{f}(x)$ back to $f(x)$, we need to apply the inverse Fourier transform:

$$f(x) = \frac{1}{\sqrt{2\pi}} \int_{-\infty}^{\infty} \hat{f}(x) e^{ikx} dk \tag{3}$$

Based on the Euler's formula, $e^{i\theta} = cos(\theta) + sin(\theta)$, the k in e^{-ikx} gives the frequencies for the sin and cos waves which is why Fourier Transform takes a function of time and turn it into a function of frequency.
Two ways to perform Fourier Transform are Discrete Fourier Transform(DFT) and Fast Fourier Transform(FFT). The most commonly used algorithm for FFT is called the Cooley–Tukey algorithm which recursively break down a DFT of size N into two DFT of size $\frac{N}{2}$. This algorithm is also known as the divide and conquer algorithm which have a time complexity of $O(Nlog(N))$ which is much faster than performing a single DFT that has a time complexity of $O(N^2)$.

2.2 Averaging

To find the frequency emitted by the submarine, we need to take the average of the data in the frequency domain and this is due to the fact that the white noise has zero mean. After averaging, the white noise will cancel out and we will have the information about the frequency that we are looking for. By the Heisenberg Uncertainty Principle, we will lose many information regarding the signal in spatial domain since averaging will give us all information of the signal in frequency space. This is the drawback of this technique.

2.3 Filtering

In the introduction, I mentioned that a filtering method will be used to remove noise from the data set. The method is called the spectral filtering where we apply a filter around the frequency that was emitted by the submarine so that all unnecessary frequencies and noises will be removed. The filter that will be used is the Gaussian filter:

$$F(k) = e^{-\tau(k-k_0)^2} \tag{4}$$

where τ is the width of the filter, k is the wavenumber, and k_0 is the center frequency that can found by averaging the Fourier Transforms. Furthermore, since we are dealing with multidimensional data, we need to make adjustment to the filter so that it can be applied to a 3-D frequency space:

$$F(k) = e^{-\tau(kx-k_{x0})^2+-\tau(ky-k_{y0})^2+-\tau(kz-k_{z0})^2} \tag{5}$$

So the only difference here is that the wavenumber and center frequency have x,y,z components.

3 Algorithm Implementation and Development

3.1 Defining domain

Before using the FFT to the data set, we need to first define the frequency domain. With the given information on time slot L and Fourier modes n, we can define the frequency domain as $(-\frac{n}{2}, \frac{n}{2} - 1)$. In addition, since FFT assumes 2π periodic signals, we need to rescale the frequency domain by $\frac{2\pi}{L}$. Because we are dealing with 3-D data, we also need to map the frequency domain onto 3-D space and we will have kx, ky, and kz as a result of calling the *meshgrid* function. This is line 2 to 16 of the code in Appendix B.

3.2 Reshaping

As I mentioned in the introduction, we need to process each column of the data into a 64×64×64 3-D matrix and an example of this process is on line 18 of appendix B. This piece of code is used every time when going through each column of the original data.

3.3 Averaging

To perform the averaging technique, we need to use a for loop that allows us to add the result of calling a multidimensional version of FFT, **fftn**, on each 64×64×64 matrix. Before taking the average of the sum, we need to use the **fftshift** function on the transformed data and this is because the FFT function in MATLAB return the positive frequency first and then the negative frequency so the **fftshift** function will give us the frequency in the correct order. Reminds that the purpose of averaging is to find the central frequency so to do that, we need to find the strongest frequency by using the **max** function. This function will return to us a maximum value and the index of the maximum value and we need to use the **ind2sub** function to map the index to the indices in kx, ky, and kz space. In the end, by plugging the indices into kx, ky, and kz, we will have the central frequency k_{x0}, k_{y0}, and k_{z0}. This is lines 22 to 38 of the code in appendix B.

3.4 Filtering

After we found the central frequency, the next step is to apply the Gaussian filter (**5**) to each transformed data around the central frequency and change back to the spatial domain by using Inverse Fast Fourier Transform(**ifftn**). With noise removed in this spatial domain, we can find the x,y,z coordinates of the submarine by finding the index of the maximum signal in each realization using the **max** function and the values returned from plugging in the index into **ind2sub** function will give us the answer. This is lines 40 to 52 of the code in appendix B.

4 Computational Results

4.1 Averaging and Filtering results

By averaging, we get k_{x0}, k_{y0}, and k_{z0} equal to 5.3407, -6.9115, 2.1991 and these values will be used to create the Gaussian filter on line 41 of appendix B. **Figure1** is a great way to visualize the result of filtering and the coordinates of the frequency that we are trying to locate is really straightforward. By filtering and finding the maximum signal, we will have a 3 by 49 matrix that stores the x, y, z coordinates of the submarine in the 24 hour time period with half-hour increments. The coordinates are displayed in $\mathbf{Table1-3}$. Reminding that in the introduction, we also aimed to find the final position of the submarine so that we can send a P-8 Poseidon subtracking aircraft to the location and in this case, the final position would be the last row of **Table3** which is $\mathbf{x = 36, y = 17, z = 54}$.

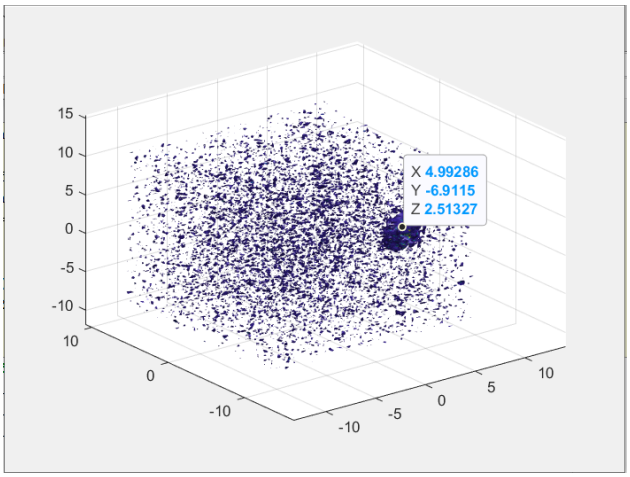

Figure 1: Isosurface plot of the resulted matrix from averaging each FFT

4.2 Path plot

Furthermore, we can plot the coordinates in $\mathbf{Table1-3}$ using the **plot3** function and the result can help us to visualize the path of the submarine. The lines 54 to 56 of appendix B are used for plotting and it results in **Figure2**.

5 Summary and Conclusions

To sum up, in this assignment, we used the averaging and filtering technique to remove the noises around the central frequency that we found. Using the denoised data, the maximum signal in the spatial domain will give us the coordinates of the submarine. In the end, we found the final location of the submarine which is $\mathbf{x = 36, y = 17, z = 54}$.

X:	33	34	35	37	38	39	40	41	42	43	44	45	46	47	48	48
Y:	43	43	43	43	43	43	43	43	43	42	42	41	40	39	39	38
Z:	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22

Table 1: The X,Y,Z coordinates of the submarine during time 0 to 7 and a half hour

X:	49	50	50	51	51	52	52	52	52	52	52	52	52	51	51	51
Y:	37	36	34	33	32	30	29	27	26	24	23	21	20	19	18	16	15
Z:	23	24	25	26	27	27	29	29	30	31	32	33	34	35	36	37	38

Table 2: The X,Y,Z coordinates of the submarine during time 8 hour to 16 hour

X:	50	49	48	48	47	46	46	44	44	43	41	40	39	38	37	36
Y:	14	14	13	12	11	11	11	11	11	11	12	12	13	14	15	17
Z:	39	40	41	42	43	44	45	46	47	48	49	49	51	51	53	54

Table 3: The X,Y,Z coordinates of the submarine during time 16 and a half hour to 24 hour

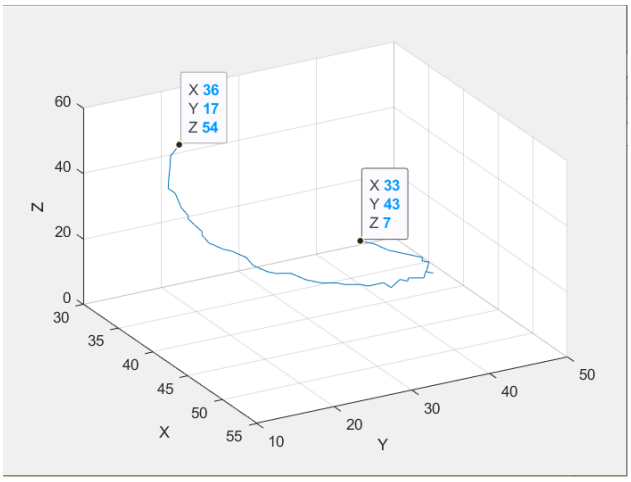

Figure 2: Path of the submarine in the 24 hour time period

Appendix A MATLAB Functions

This is the list of MATLAB function that was mentioned in this assignment(most definitions are from the MathWork website):

- `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.
- `B = reshape(A, sz1,...,szn)` reshapes the matrix A into a sz1 by sz2 by ,,,, by szn matrix.
- `Y = fftshift(X)` rearranges a Fourier transform X by shifting the zero-frequency component to the center of the array.
- `[M,I] = max(A)` return a maximum value M in the matrix A and the corresponding index of M.
- `Y = fftn(X)` returns the multidimensional Fourier transform of an N-D array X using a fast Fourier transform algorithm.
- `[I1,I2,...,In] = ind2sub(sz,ind)` returns n arrays I1,I2,...,In containing the equivalent multidimensional subscripts corresponding to the linear indices ind for a multidimensional array of size sz.

- `X = ifftn(Y)` compute the Inverse Fourier Transform of Y using Fast Fourier Transform.
- `plot3(X,Y,Z)` plot multiple sets of coordinates on the same set of axes in 3-D space.
- `isosurface(X,Y,Z,V,isovalue)` plot and computes isosurface data from the volume data V at the isosurface value specified in isovalue.

Appendix B MATLAB Code

Add your MATLAB code here. This section will not be included in your page limit of six pages.

```
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);
for j=1:49
   Un(:,:,:)=reshape(subdata(:,j),n,n,n);
    M = max(abs(Un),[],'all');
end
%% Averaging
avg = zeros(64,64,64);
for j=1:49
   Un(:,:,:)=reshape(subdata(:,j),n,n,n);
   Unt = fftn(Un);
   avg =  avg + Unt;
end
avg = abs(fftshift(avg))./49;
[mxv,idx] = max(avg(:));
[x1,y1,z1] = ind2sub(size(avg),idx);
%% Generating Figure 1
avg = abs(avg)/max(abs(avg(:)));
isosurface(Kx,Ky,Kz,avg,0.2);
axis([-20 20 -20 20 -20 20]), grid on, drawnow
%% Center frequency
Kx0 = Kx(x1,y1,z1);
Ky0 = Ky(x1,y1,z1);
Kz0 = Kz(x1,y1,z1);
%% Gaussian Filter
tau = 0.8;
filter = exp(-tau*(Kx - Kx0).^2 + -tau*(Ky - Ky0).^2 + -tau*(Kz - Kz0).^2);
%% X,Y,Z coordinates of the submarine
res = zeros(3,49);
for j=1:49
   Un(:,:,:)=reshape(subdata(:,j),n,n,n);
   Unt = fftshift(fftn(Un));
   Untf = Unt.*filter;
   Unf = ifftn(Untf);
   [m,i] = max(Unf(:));
   [x2,y2,z2] = ind2sub(size(Unf),i);
   res(:,j) = [x2,y2,z2];
end
%% Plotting the result
plot3(res(1,:),res(2,:),res(3,:));
grid on, drawnow
xlabel('X'); ylabel('Y'); zlabel('Z');
```


Listing 1: Code for this assignment