AMATH482 Homework 1

Vandy Zhang
January 25, 2021

Abstract

This assignment is to use algorithms to denoise the signals detected by a technology and to locate the path of a submarine by implementing the advanced Fourier Transform in MATLAB. Therefore, we can get the specific horizontal locations of the plane that is going to track the submarine based on the path of the submarine we analyzed.

1 Introduction and Overview

For this homework, I was asked to hunt for a submarine, find its trajectory under the sea, and figure out the positions I should send my P-8 Poseidon aircraft to sub track the submarine. A new submarine technology was introduced to detect acoustic frequency emitted by a submarine. I was offered with a noisy acoustic dataset subdata.mat (a dataset that has 262144 rows and 49 columns) which contains 49 measurements of the frequency of the submarine over a 24-hour span at half-hour increments in time. Each column of the data has 262144 data points representing the acoustic frequency with noise.

To locate the positions of the submarine at these 49 time points, I first reorganized the subdata.mat and put the 262144 data points in the $64 * 64 * 64$ dimension. I then applied the Fourier Transform to convert the data of space to data of frequency. By averaging the spectrum, I found out the center frequency over the 49 measurements generated by the submarine, which finishes the part one of the homework. Then I was asked to filter the data around the center frequency and determine the path of the submarine. I applied the 3-Dimensional Gaussian filter function and filtered the data around the center frequency I got in part 1 over time domain. Then I used some MATLAB functions to locate the maximum points of the filtered frequency and get their coordinates. The path of the submarine was determined by the 49 3-dimensional data points.

2 Theoretical Background

Since the subdata.mat provided with a 262144 * 49 (space by time) matrix, I have to use Fourier Transform to convert the data in space to data in frequency so that I can better analyze the acoustic frequency of the submarine. Here is the Fourier Transform Equation [2]:

Suppose we are given a function f(x) with x ∈ R. We define the Fourier Transform of f(x), written as

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

.

In this case, we have our data over frequency domain. Theoretically, we are making projections of the frequencies over time to one time point. However, we will lose time information this way. To have frequencies over time domain, the Inverse Fourier Transform will work. This also helps if I am given f_hat and want to recover my f(x). Here is the Inverse Transform Equation [2]:

Suppose we have the same condition as the Fourier Transform above. The Inverse Fourier Transform Equation can be written as

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

If we are given a function at a discrete set of points, it's possible that we could lose some high frequency values occurred between those points. The Discrete Fourier Transform truncates at some maximum frequency so that it can reflect this potential shortcoming [2].

Suppose we are given a sequence of N values that form a function sampled at equally-spaced points $\{\mathrm{x}_0, \mathrm{x}_1, \ldots, \mathrm{x}_{\mathrm{n}-1}\}$. The Discrete Fourier Transform is a sequence given by

$$\mathrm{x_{hatk}} = \frac{1}{N} \sum_{n=0}^{N-1} x_n e^{\frac{2\pi i k n}{N}}$$

The Fast Fourier Transform is a function that does the Discrete Fourier Transform faster. It helps split the DFT into two DFT sequences and use a divide and conquer algorithm [2]. The Discrete Fourier Transform has a complexity of $\mathcal{O}(\mathrm{N}^2)$, while the Fast Fourier Transform only has a complexity of $\mathcal{O}(\mathrm{N}\log(\mathrm{N}))$. The built-in function in MATLAB for two-dimensional Fast Fourier Transform is fft (fftn for multidimensional FFT). The built-in function in MATLAB for two-dimensional Inverse Fast Fourier Transform is ifft (ifftn for multidimensional IFFT).

3 Algorithm Implementation and Development

When setting up the coordinates for the frequency domain, I set up **k** by implementing 2 * [0: (N/2 – 1) – N/2 : -1]. I rescaled the frequencies by 2 π/L since my original data has L for its special domain, but FFT assumes 2 π periodic signals. Because of aliasing in the MATLAB implementation, I had to **fftshift** my **k** so that I could have it in a correct order. Since the acoustic frequency data of the submarine is in 3-Dimension, I implemented **meshgrid** to properly set the 3-Dimensional coordinates up.

To calculate the center frequency generated by the submarine, I looped the 49 sets of data and used **reshape** method to have each set converted to 64 * 64 * 64 dimension. Within each iteration, I used the Fast Fourier Transform for multidimensional data, which is the **fftn** method in MATLAB for each set of my data and added the transformed 49 measurements of frequencies up. By averaging the measurements and using the **fftshift** method to correct the order, I got the mean of the frequency data in hand. The center frequency, which refers to the actual strike of the signal would be the maximum point of the mean data. I also used the **max** method to access the maximum value of the frequency data and returned the index of the maximum value to locate its x, y, z coordinates. Then I used **ind2sub** to convert the linear index to multidimensional subscripts and then got the actual x, y, z coordinates of the center frequency.

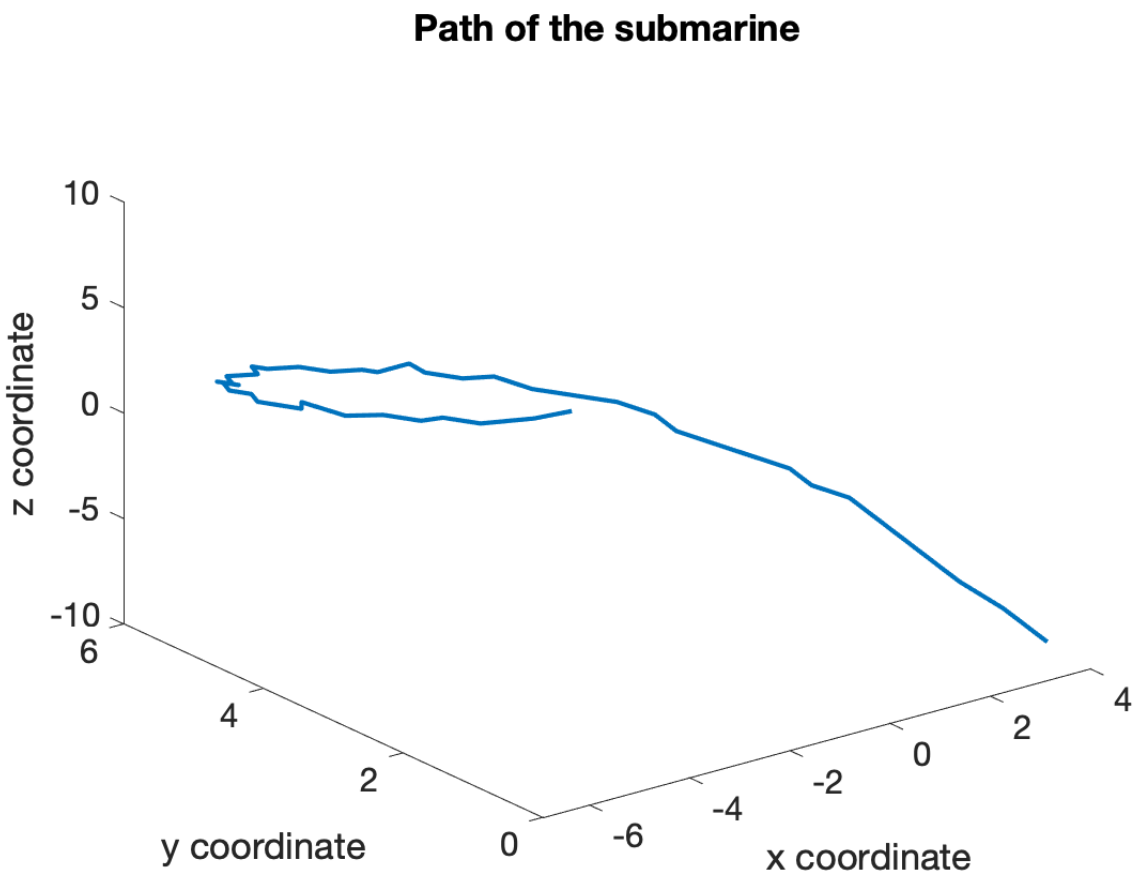
I used the 3-Dimensional Gaussian Filter function to denoise the data around its center frequency. I repeated the process of making a loop, reshaping the dimensions, and using the Fast Fourier Transform to get my data over frequency domain. Then I applied my 3D Gaussian filter to filter the noise. After that, I implemented the **ifftshift** method to undo the result of **fftshift** and reorganized my data. I then used the Inverse Fast Fourier Transform to convert my multidimensional filtered data back to time domain by applying the **ifftn** method in MATLAB. To actually get the coordinates of the exact location of the submarine, I again found the maximum value by using **max** and implemented the **ind2sub** method to convert the linear index to actual x, y, z coordinates. Having the 49 coordinates, I used **plot3** to connect them and form a path in the 3D space.

4 Computational Results

4.1 Center Frequency

The center frequency of the acoustic signals generated by the submarine is at the point (5.3407, -6.9115, 2.1991).

4.2 The Path of the Submarine



4.3 Table of the X, Y Coordinates

I should send my P-8 Poseidon subtracking aircraft to follow the coordinates listed in the table below over the 24-hour span.

#	1	2	3	4	5	6	7	8	9	10
x	3.1416	3.1416	3.1416	3.1416	3.1416	3.1416	3.1416	3.1416	3.1416	2.8274
y	0	0.3142	0.6283	1.2566	1.5708	1.8850	2.1991	2.5133	2.8274	3.1416
#	11	12	13	14	15	16	17	18	19	20
x	2.8274	2.5133	2.1991	1.8850	1.8850	1.5708	1.2566	0.6283	0.3142	0
y	3.4558	3.7699	4.0841	4.3982	4.7124	5.0265	5.0265	5.3407	5.3407	5.6549
#	21	22	23	24	25	26	27	28	29	30
x	-0.6283	-0.9425	-1.2566	-1.8850	-2.1991	-2.8274	-3.1416	-3.4558	-4.0841	-4.3982

y	5.6549	5.9690	5.9690	5.9690	5.9690	5.9690	5.9690	5.9690	5.9690	5.9690
#	31	32	33	34	35	36	37	38	39	40
x	-4.7124	-5.3407	-5.6549	-5.9690	-5.9690	-6.2832	-6.5973	-6.5973	-6.9115	-6.9115
y	5.6549	5.6549	5.3407	5.3407	5.0265	5.0265	4.7124	4.3982	4.0841	3.7699
#	41	42	43	44	45	46	47	48	49	
x	-6.9115	-6.9115	-6.9115	-6.5973	-6.2832	-6.2832	-5.9690	-5.3407	-5.0265	
y	3.4558	3.4558	2.8274	2.5133	2.1991	1.8850	1.5708	1.2566	0.9425	

5 Summary and Conclusions

Hunting for the submarine, I applied the Fast Fourier Transform to convert the original data to 3-D data over the frequency domain. The peak of the mean frequency was the center frequency I was finding. The 3-D Gaussian filter function helped me denoise the data over the frequency domain, and the inverse Fast Fourier Transform helped me to transfer the data back to the domain of time. My data of the path of the submarine was then generated. Since my P-8 Poseidon subtracking aircraft would be tracking the submarine at the exact horizontal position the submarine was going to, the positions of the aircraft we were sending to could be the ones of the submarine.

References

[1] Jose Nathan Kutz. Data-driven modeling & scientific computation: methods for complex systems & bigdata. Oxford University Press, 2013.
[2] University of Washington. AMATH482 Lecture 1 Notes. URL: https://canvas.uw.edu/courses/1433289/files/72729571?wrap=1

Appendix A. MATLAB Functions

load Load variables from a file into MATLAB workspace
linspace (x1, x2, n) Generate linearly spaced vector between X1 and X2 with a length of n.
meshgrid (x, y, z) Generate 3-D grids
reshape (A, sz) Reshape A to have size sz
fftn (X) Apply the N-D Fast Fourier Transform to X
fftshift (X) Shift zero-frequency component to the center of the spectrum
ifftn (X) Apply the N-D Inverse Fast Fourier Transform to X
ifftshift (X) Inverse the zero-frequency shift conducted by fftshift
zeros (n) Create array of all zeros with size n

max (X) Find the maximum elements in X

ind2sub (sz, ind) Convert the linear indices in ind to subscripts in a dimension with size sz

isosurface (X, Y, Z, V, isovalue) Compute the data from V at specified isovalue and produce a multidimensional surface in x, y, z coordinates

plot3 (X, Y, Z) Plot lines in 3-D space

Appendix B. MATLAB Code

```
% 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);

Utave  = zeros(1, n);

for j=1:49
Un(:,:,:)=reshape(subdata(:,j),n,n,n);
Ut(:,:,:) = fftn(Un(:,:,:));
Utave = Utave + Ut(:,:,:);
% 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

sz = [64 64 64];
Utave = fftshift(Utave)/49;
[M, I] = max(abs(Utave(:)),[],'all','linear');
[x0, y0, z0] = ind2sub(sz, I);
center(1) = Kx(x0, y0, z0);
center(2) = Ky(x0, y0, z0);
center(3) = Kz(x0, y0, z0);
% isosurface(X,Y,Z,abs(Utave)/max(abs(Utave), [], 'all'),0.7)

% Problem 2
tau = 0.2;
x0 = center(1); y0 = center(2); z0 = center(3);
filter = exp(-tau*((Kx - x0).^2 + (Ky - y0).^2 + (Kz - z0).^2)); % Define the
filter
indice = zeros(1);
sz = [64 64 64];
```

```
for i = 1:49
Un(:,:,:)=reshape(subdata(:,i),n,n,n);
Ut(:,:,:) = fftshift(fftn(Un(:,:,:)));
unft = filter.*Ut;
unf = ifftn(ifftshift(unft));
[M, indice(i)] = max(abs(unf(:)),[],'all','linear');
end

[xvector, yvector, zvector] = ind2sub(sz, indice);
path_x = Kx(xvector, yvector, zvector);
path_y = Ky(xvector, yvector, zvector);
path_z = Kz(xvector, yvector, zvector);

figure(2)
plot3(squeeze(path_x(1,:,1)), squeeze(path_y(:,1,1)), squeeze(path_z(1,1,:)),
'LineWidth', 2);
xlabel("x coordinate")
ylabel("y coordinate")
zlabel("z coordinate")
title("Path of the submarine")
set(gca, "fontsize", 15);
print('HW1Path.png', '-dpng');
```