AMATH 582 Homework 1

Joseph David

January 27, 2020

Abstract

In this project, we are given (white) noisy acoustic data over a fixed 3D underwater region, over 49 30-minute intervals spanning 24 hours. A submarine in our region has a specific acoustic frequency, and our goal is to find the position of the submarine at each of the 49 instances. We do this by running `Matlab` code that computes the Fast Fourier Transform (FFT) of our data to determine the frequency signature and average it over time, which we use to find the submarine's acoustic signature. Then, we filter the FFT of the acoustic data around said frequency and take the inverse Fourier Transform to determine the position where this signature is being detected, ie. the submarine's position.

1 Introduction and Overview

Acoustic data in a 3D space is given at 49 intervals. We are initially given a 262144-by-49 matrix `subdata`, in which each of the 49 columns is the acoustic data over one interval. Note that $262144 = 64 \times 64 \times 64$, so we reshape each column of `subdata` into a 64x64x64 matrix. The 3D space is 20x20x20 and represents a region underwater. Thus, we divide the x, y, and z axes into 64 equal parts and each realization of `subdata` represents an amplitude at that point in our region. However, this data is noisy due to signal pollution, and hence we cannot determine the position of the submarine directly from `subdata`. Therefore, averaging the Fourier Transform of each realization has the effect of cancelling out the noise while continually adding the submarine's frequency to itself. We therefore see that the largest value in the frequency domain will be that of the submarine. Once we have the submarine's frequency, we filter around that frequency, which isolates it, and then take the inverse Fourier Transform to find the submarine's position.

2 Theoretical Background

The **Fourier Transform** (FT) operator maps functions on a spatial domain to functions on a frequency domain. The fundamental idea behind the FT is that a generic function $f(x)$ may be represented by

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

where the function $\widehat{f}(k)$ is called the Fourier Transform of $f(x)$. Thus, $f(x)$ may be represented by a "sum" of trigonometric functions e^{ikx}. The value $\widehat{f}(k)$ is the amplitude, or amount, of the wave with wavelength $2\pi/k$ in the function $f(x)$. To compute $\widehat{f}(k)$ we use the formula

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

Notice that (1) gives a way to compute $f(x)$ given its FT $\widehat{f}(k)$. For this reason, (1) is called the Inverse Fourier Transform.

In order to find the submarine's frequency, we assume that the noise in the signal is random. This implies that if we average over sufficiently many signal realizations, the noise will cancel itself out leaving only the frequency we are interested in. If the noise were not random, then averaging would not necessarily isolate our frequency.

3 Algorithm Implementation and Development

The basic steps are twofold:

1. Identify the submarine's acoustic frequency by averaging the FT of the acoustic data over the 49 measurement intervals.
2. Filter the FT at each instance and compute the Inverse FT.

Algorithm 1: Determine Submarine Frequency

Import subdata from `subdata.mat`
for $j = 1 : 49$ **do**
  Extract the jth acoustic measurement from `subdata`
  Take its FT `Ut`
  Add it to the existing average `Utave`
end for
Find the indices of the largest element of `Utave`

To implement Algorithm 1, we use the `reshape` command to convert the columns of `subdata` into 64x64x64 arrays. Then we use `fftn` to compute the Fourier Transform of each realization and add it to the store average variable. After all 49 realizations have been averaged, we use `max` to find the indices of the maximal element of the average Fourier Transform.

Algorithm 2: Determine Submarine Path

Import subdata from `subdata.mat`
Create filter centered around submarine frequency
for $j = 1 : 49$ **do**
  Extract the jth acoustic measurement from `subdata`
  Take its FT, `Ut`
  Multiply `Ut` by filter to isolate submarine frequency
  Take inverse FT, `U`
  Find indices of largest element of `U` and store it, this will be the submarine's position at the jth point in time
end for
Plot path of submarine

To implement Algorithm 2, we first use the indices obtained in Algorithm 1 to create `filter` which is a *sech* function centered at those indices. Then, as in Algorithm 1, we take each realization of `subdata` and take its Fourier Transform. However, we now multiply it by `filter` and then use `ifftn` to compute the inverse Fourier Transform. Using `max` once again, we determine the indices of the maximal element, which will be the position of the submarine. We store each of these in a vector, which we then plot using `plot3`.

4 Computational Results

See Figures 1 and 2 below.

5 Summary and Conclusions

We were successfully able to track the submarine's position. Averaging was a very effective way of denoising the data and to recover the true signal. We found that the frequency of the submarine was roughly $(5.34, -6.91, 2.20)$. After filtering around this frequency and taking the inverse Fourier Transform, we plotted

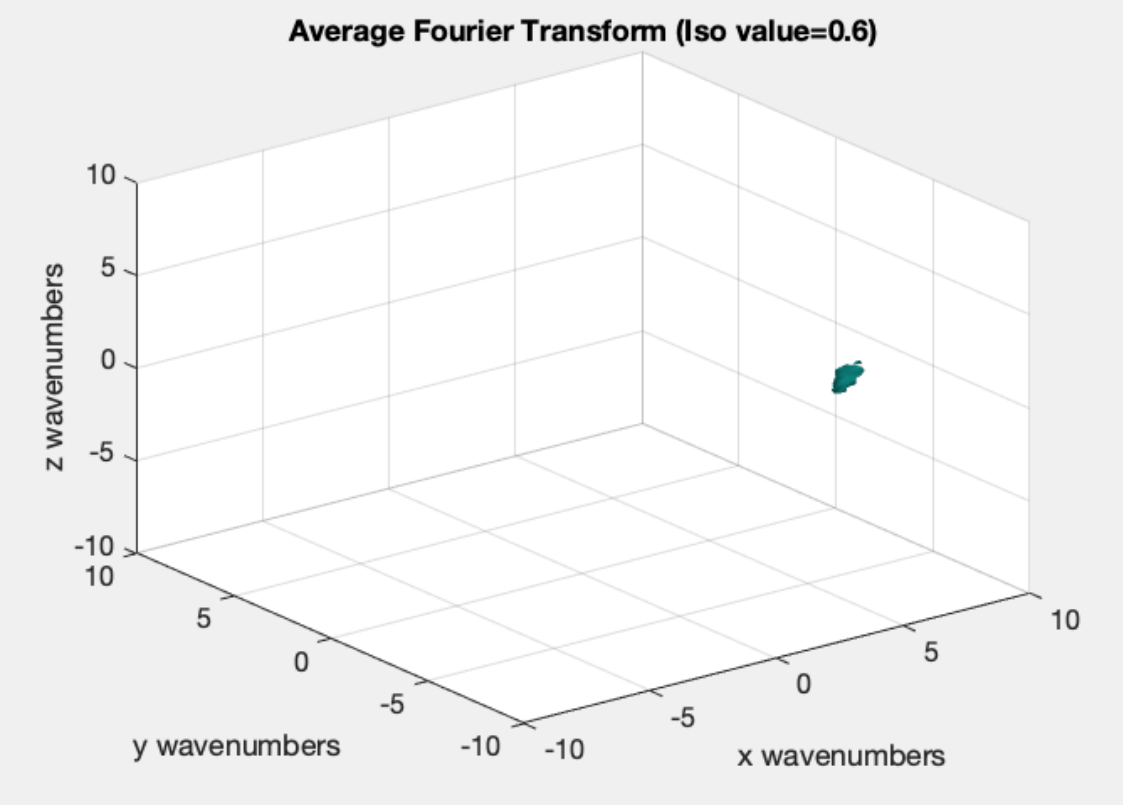

Figure 1: Isosurface plot of the average Fourier Transform with isosurface value = 0.6

the position of the submarine at each of the 49 points in time as a line plot. We saw that the submarine started out at $(3.125, 0, -8.125)$ and ended at $(-5, 0.94, 6.56)$

Appendix A MATLAB Functions

- `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.
- `g = fftn(f)` returns the mulitdimensional Fourier Transform of f, where f is a multidimensional array.
- `y = fftshift(x)` centers the matrix `x` around the zero frequency. It is commonly used after computing the Fourier Transform, so that your data is centered around the zero frequency.
- `[M,I] = max(A,[],'all','linear')` returns the maximal value `M` of the multidimensional array `A` along with the *linear* index `I` where the maximum is attained.
- `[I1,..., In] = ind2sub(sz, I)` converts the linear index I to its indices along each dimension, where `sz` is an array that contains the size of the multidimensional array.
- `isosurface(X,Y,Z,f,v)` plots the surface $f = v$, where `X`, `Y`, `Z`, and `f` are n-by-n-by-n arrays containing the x, y, and z coordinates and f is the function values, respectively.
- `plot3(x,y,z)` takes in three coordinate vectors `x`, `y`, and `z` of size n and returns a line plot following the n points with coordinates determined by `x`, `y`, and `z`.

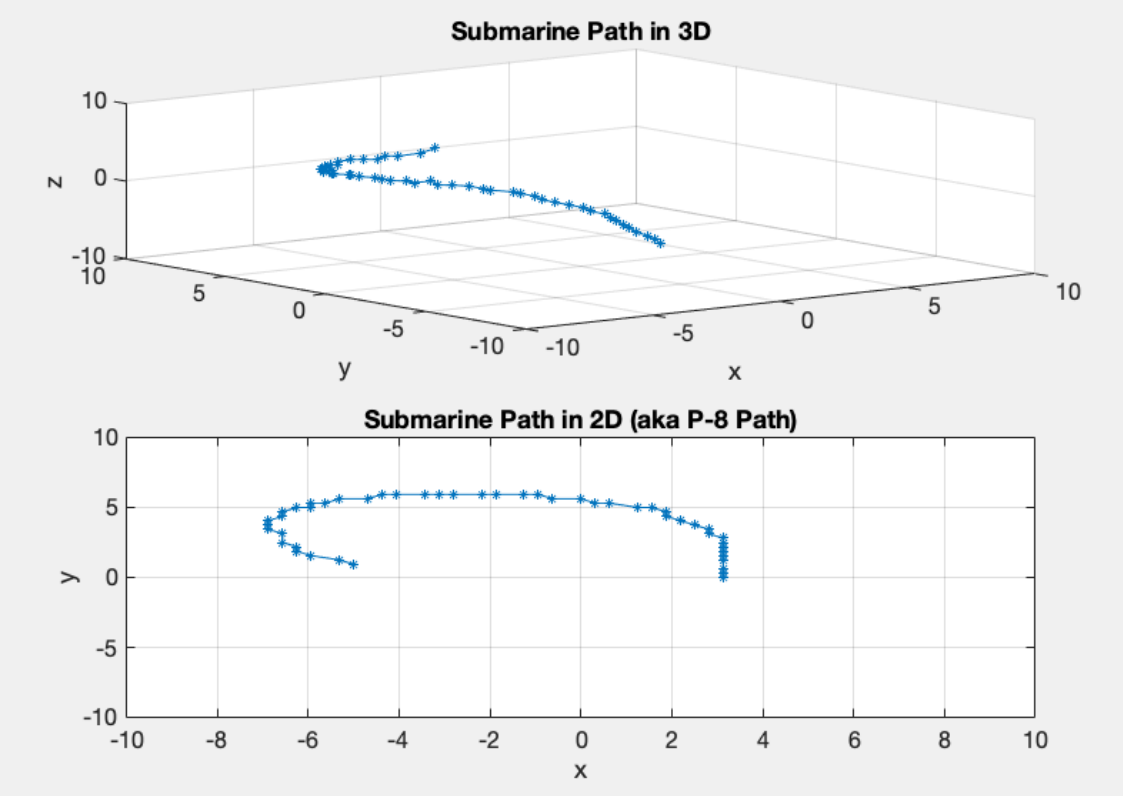

Figure 2: Plot of the Submarine Path

Appendix B MATLAB Code

See code below.

```
clear all; close all; clc
load subdata.mat; % imports the data as the 262144 x 49 (space by time) matrix
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);

%first, average over the frequencies to find the center frequency
Unfave = zeros(n,n,n);
for j = 1 : 49
    Un = reshape(subdata(:,j),n,n,n);
    Utn = fftshift(fftn(Un));
    Unfave = Unfave + Utn;
end
[M,I] = max(abs(Unfave), [], 'all', 'linear');

figure(1)
isosurface(Kx,Ky,Kz, abs(Unfave)/M, 0.6), grid on
title('Average Fourier Transform (Iso value=0.6)');
xlabel('x wavenumbers'); ylabel('y wavenumbers'); zlabel('z wavenumbers');
ax = gca; ax.FontSize = 16; xlim([-10 10]); ylim([-10 10]); zlim([-10 10]);

sz = [n n n];
[I1, I2, I3] = ind2sub(sz,I); %contains the indices of the max element in Unfave

%create filter centered around the center frequency, center frequency is at (ks(I2), ks(I1), ks(I3))
c = .9; filter = sech(c*(Kx - ks(I2))) .* sech(c*(Ky - ks(I1))) .* sech(c*(Kz - ks(I3)));

%go through each realization of subdata and Fourier transfrom
%then apply filter and take inverse Fourier
a = zeros(1,49);
b = zeros(1,49);
c = zeros(1,49);
for j = 1 : 49
    Un = reshape(subdata(:,j),n,n,n);
    Utn = fftshift(fftn(Un));
    Utnf = Utn .* filter;
    U = ifftn(Utnf);
    [M2, J2] = max(abs(U),[],'all', 'linear');
    [b(j), a(j), c(j)] = ind2sub(sz, J2);
end

figure(2),
subplot(2,1,1), plot3(x(a), y(b), z(b), '-*'), grid on;
title('Submarine Path in 3D'); xlabel('x'); ylabel('y'); zlabel('z');
ax = gca; ax.FontSize = 16; xlim([-10 10]); ylim([-10 10]); zlim([-10 10]);
subplot(2,1,2), plot(x(a), y(b), '-*'), grid on;
title('Submarine Path in 2D (aka P-8 Path)'); xlabel('x'); ylabel('y');
ax = gca; ax.FontSize = 16; xlim([-10 10]); ylim([-10 10]);
```

Listing 1: Matlab code for this project.