AMATH 482 Homework 1

Erik Huang

January 27, 2020

Abstract

Experiment how to analyze noisy acoustic data from a submarine using Fast Fourier Transform, and how to denoise data to expose the frequency signature by using averaging and filtering.

1 Introduction and Overview

Our goal is to determine the position of the submarine in the Puget Sound using the provided noisy acoustic data. The acoustic data from the submarine is collected over a 24-hour period in half-hour increments, using a broad spectrum recording of acoustics. Since it's a new submarine technology, we don't know the acoustic frequency that emits from the submarine. Yet, we can average the spectrum to get the frequency signature (center frequency).

Once we obtain the frequency signature of the submarine, we can then try to locate the submarine and find its trajectory using the data collected over the span of 24-hours.

2 Theoretical Background

2.1 Fast Fourier Transform

In this assignment, we need to use Fourier Transform to transform the received acoustic data from spatial domain to frequency domain so that we can analyze the frequency signature.

The Fast Fourier Transform (FFT) is an algorithm that essentially does the Discrete Fourier Transform (DFT) faster and more efficient. To understand its functionality, we can look at the DFT. The DFT is a variation of Fourier Transform which is used when given a discrete set of points. Rather than have a range for x in a continuous set of points, a discrete set of points contains N distinct and equally-spaced points of x_n: $\{x_0, x_1, ...x_{N-1}\}$. The result of DFT is a sequence of numbers given by

$$x_n = \frac{1}{N} \sum_{k=0}^{N-1} X_k e^{2\pi i k n / N}.$$

Adapted from the DFT, the FFT improves the run time from $O(N^2)$ to $O(Nlog(N))$.

2.2 Averaging of the Spectrum

The reason why we need to average the spectrum after doing the FFT is that doing so can help us get rid of the white noise in the data. Figure 1 and figure 2 demonstrates the effect of the averaging of the data in the frequency domain. We can clearly see that the cluster of white noise frequency has been taken out successfully in the figure 2. Since white noise can be treated as a normally distributed random variable with zero mean, it will become zero when taking the average.

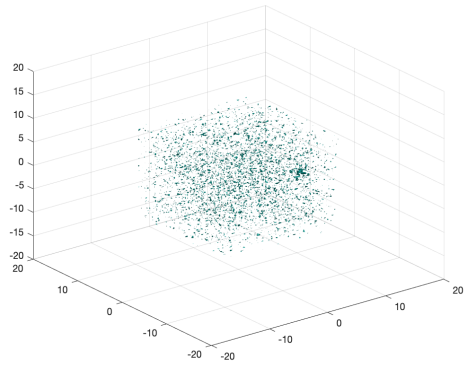

Figure 1: Before averaging in frequency domain

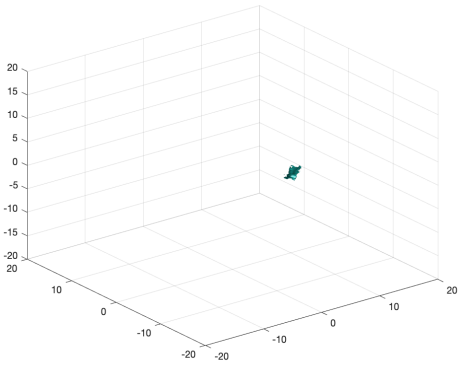

Figure 2: After averaging in frequency domain

2.3 Filter function

For the filtering of data in this assignment, we will use a simple function introduced in class:

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

This is a Gaussian function with τ determining the width of the filter and the constant k_0 determining the centre of the filter. The way we use filter function is simply multiply all the signals in the frequency domain by this function. Doing so can help us expose the pattern and location of our desired frequency.

3 Algorithm Implementation and Development

3.1 Averaging of the spectrum and determine the frequency signature

The provided acoustic data, "subdata.mat", is in a form of a 262144×49 matrix. In order to be able to plot it, we first need to reshape each entry into a $64 \times 64 \times 64$ vector. We can do this task in an iteration. During this first iteration, we will do two more tasks: Fast Fourier Transform (FFT) and calculate the sum.

Algorithm 1: Preprocessing and FFT

```
Import data from subdata.mat
for j = 1 : 49 do
    reshape entry j from subdata
    Run FFT on the reshaped vector
    Add the reshaped vector to the sum
end for
```

The order of these tasks cannot be switched. The FFT process transfers the data from the time domain into the frequency domain, which is really important for the averaging process. Since averaging the data in the time domain is not helpful due to the movements of the submarine. After the iteration, we now have a sequence of data in the frequency domain as well as its sum, which will then be used to calculate the average.

To obtain the $\vec{\mu}$, we can divide the absolute value of the sum by the length of the sequence. Note that in the MATLAB code we need to run `fftshift` function on the transformed vectors because the index k for x_k needs to be shifted in order.

Once we have the averaged result, we can locate the center frequency by finding the occurrence of the maximum frequency. This can be done by using the default `max` function within the MATLAB, which returns the index position of the occurrence.

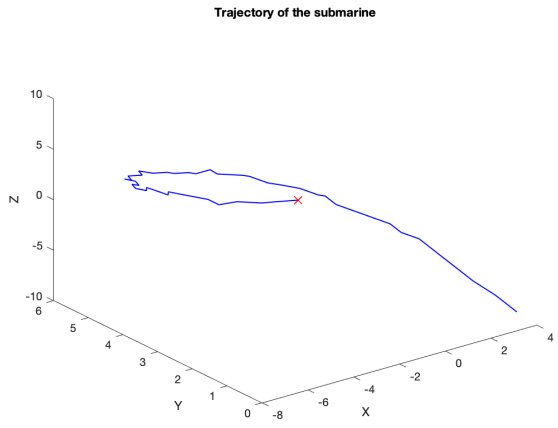

Figure 3: Trajectory of the submarine

3.2 Filter the data around the center Frequency

As mentioned earlier, we will use this filter function:

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

However, since this filter function doesn't apply to the dimensions of the transformed vectors, we need to rewrite the function as:

$$F(k) = e^{-\tau\left((k-k_{x_0})^2+(k-k_{y_0})^2+(k-k_{z_0})^2\right)}$$

In addition, we need to rescale our k by $2\pi/L$ since the FFT assumes 2π periodic signals.

Once we define this function in the MATLAB code, we can apply the filter to the data around the center frequency. This process needs to be done in a for loop, similar to the first algorithm. Yet in this algorithm, we don't need to do the averaging task, instead, we apply the filter function to each entry in the frequency domain and finally transfer all entries back into the time domain. Because of the filter we applied, the amplified signal can then be detected during each time interval.

Algorithm 2: Filtering the frequency

$filter = exp(-tau * ((K_x - x_0)^2 + (K_y - y_0)^2 + (K_z - z_0)^2))$;
for $j = 1 : 49$ **do**
  reshape entry j from `subdata`
  $utn = fftn(entryj)$;
  $filteredV = filter. * utn$
  $unf = ifft(filteredV)$;
  Locate the position of the maximum signal
  Record the position as submarine's trajectory
end for

4 Computational Results

After running the above algorithm, we can plot the data we gathered during the iteration, which corresponds to the position of the submarine at each time interval. The x and y coordinates are listed in the table below.

	X Position	Y Position
1	3.125	0
2	3.125	0.3125
3	3.125	0.625
4	3.125	1.25
5	3.125	1.5625
6	3.125	1.875
7	3.125	2.1875
8	3.125	2.5
9	3.125	2.8125
10	2.8125	3.125
11	2.8125	3.4375
12	2.5	3.75
13	2.1875	4.0625
14	1.875	4.375
15	1.875	4.6875
16	1.5625	5
17	1.25	5
18	0.625	5.3125
19	0.3125	5.3125
20	0	5.625
21	-0.625	5.625
22	-0.9375	5.9375
23	-1.25	5.9375
24	-1.875	5.9375
25	-2.1875	5.9375
26	-2.8125	5.9375
27	-3.125	5.9375
28	-3.4375	5.9375
29	-4.0625	5.9375
30	-4.375	5.9375
31	-4.6875	5.625
32	-5.3125	5.625
33	-5.625	5.3125
34	-5.9375	5.3125
35	-5.9375	5
36	-6.25	5
37	-6.5625	4.6875
38	-6.5625	4.375
39	-6.875	4.0625
40	-6.875	3.75
41	-6.875	3.4375
42	-6.875	3.4375
43	-6.875	2.8125
44	-6.5625	2.5
45	-6.25	2.1875
46	-6.25	1.875
47	-5.9375	1.5625
48	-5.3125	1.25
49	-5	0.9375

Table 1: Position changes of the submarine

5 Summary and Conclusions

From this assignment, we can see how FFT can be the foundation for data analysis in these scenarios. Only with the information in the frequency domain, we could use other techniques such as averaging and filtering on the data. We can also see how effective the averaging technique is when it comes to getting rid of the white noise in acoustic data. Lastly, the filter function is the last key to resolve the position of the submarine. When combined with the `ifftn` function from MATLAB, it becomes super powerful in amplifing the signal.

Appendix A MATLAB Functions

Major Functions list:

- `load` Reads data from the local directory.
- `reshape` Changes the dimension of the array
- `fftn` N-D Fast Fourier Transform
- `fftshift` Shift zero-frequency component to center of spectrum.
- `isosurface` Extract isosurface data from volume data
- `ifftn` Inverse Fast Fourier transform on data to bring it back to the original domain.

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 5

%% FFT
L = 10; % spatial domain
n = 64; % Fourier modes

x2 = linspace(-L,L,n+1); x = x2(1:n); y = x; z = x;
i = (2*pi/(2*L))*[0:(n/2 - 1) -n/2:-1]; ks = fftshift(i);

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

sum = zeros(n,n,n);
for j=1:49
    Un(:,:,:)=reshape(subdata(:,j),n,n,n);
    Utn = fftn(Un);
    sum = sum + Utn;
end


%% Display data before Averaging
isosurface(X, Y, Z, fftshift(abs(Utn)) / max(abs(Utn),[],'all'), 0.5)
axis([-20 20 -20 20 -20 20]), grid on
drawnow

%% Average the spectrum to cancel out white noise and find the center frequency
avg = abs(fftshift(sum))/49;
[M,I] = max(avg(:));
cf = [Kx(I),Ky(I),Kz(I)]; % Location of the center frequency
isosurface(Kx, Ky, Kz, avg / max(avg(:)), 0.5)
axis([-20 20 -20 20 -20 20]), grid on
drawnow

```

```
%% filter function
tau = 1;
filter = exp(-tau*((Kx-cf(1)).^2+(Ky-cf(2)).^2+(Kz-cf(3)).^2));

traj_x = zeros(1,49);
traj_y = zeros(1,49);
traj_z = zeros(1,49);

for i = 1:49
    Un(:,:,:) = reshape(subdata(:,i),n,n,n);
    filt = fftn(Un).* fftshift(filter);
    filt = fftshift(filt);
    unfilt = ifftn(filt);
    unfilt = unfilt / max(unfilt(:));
    [M,I] = max(unfilt(:));
    cf = [X(I),Y(I),Z(I)];
    traj_x(1,i) = cf(1);
    traj_x(1,i) = cf(2);
    traj_z(1,i) = cf(3);
end

plot3(traj_x, traj_y, traj_z, 'b', 'LineWidth', 1)
hold on
plot3(traj_x(49), traj_y(49), traj_z(49), 'rx', 'Markersize', 5)
xlabel('X');
ylabel('Y');
zlabel('Z');
title("Trajectory of the submarine");
```