AMATH 482 Homework 1

Jeremy Lu

January 27, 2021

Abstract

We are looking for a submarine that is lost in the Puget Sound. Given noisy 3-D acoustic data with an unknown frequency, we want to detect the path and location of the submarine. Using MATLAB in this project, we first average the spectrum to detect the frequency domain. With this, we can denoise our data using a 3-D Gaussian filter and then plot the trajectory and final position of the submarine.

1 Introduction and Overview

When collecting signal and frequency data, rarely do we have perfect and clean results which can be analyzed immediately; instead, the data often comes with extraneous white noise. However, this can be remedied through both averaging and filtering, which enables us to focus in on the data we actually want to look at it.

In this project's case, our data is acoustic frequency emitted from a submarine. We want to locate the submarine, as well as its path. Implemented in MATLAB, we first using a Fourier Transform and averaging, we can find the central frequency in our data. We then apply a Gaussian filter on this central frequency to remove the noise and focus only on the submarine's frequency. With the data being devoid of noise, it is now possible to find the position and trajectory of the submarine.

2 Theoretical Background

2.1 Fourier Transform

The Fourier Transform is an important tool that allows any function $f(x)$ to be written as the sum of sines and cosines. In the domain $x \in [-\infty, \infty]$, the this can be written as:

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

Additionally, the Inverse Fourier Transform is defined as:

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

Essentially, the Fourier Transform is able to decompose a signal in the time domain into its frequency components. However, this comes at a cost—a narrow signal in the time domain will have a wide spread in the Fourier domain, and vice versa.

Note that the original Fourier Transform is an integral over infinity, which takes nontrivial processing power to compute using software such as MATLAB. Instead, the Fast Fourier Transform (FFT) is employed as it operates under $O(n \log n)$ time, which is ideal for large data. The FFT is able to run so quickly by discretizing $x \in [-L, L]$ with 2^n points. Two key characteristics of the FFT are that it centers on zero frequency and that it works in a domain of period 2π. Because FFT shifts the data to fit this zero frequency requirement, it is important to shift it back in order to gain meaningful frequency data. Additionally, it is necessary to rescale the frequency data by multiplying it with $\frac{2\pi}{L}$ in order to fit this requirement.

2.2 Signal Averaging

Signal Averaging utilizes the fact that the noise in the data is white noise, which will have a mean of zero. Applied in the time domain, averaging over each of the signals the noise will be greatly reduced, while also taking a more uniform distribution. In the context of this project, performing Signal Averaging will make the submarine frequencies stand out, thus allowing us to find the central frequency

2.3 Gaussian Filtering

Filtering is an important aspect of signal processing, as it allows us to ignore unnecessary noise and data. This process improves the ability to detect a signal. The Gaussian filter is defined as

$$F(x) = exp(-\tau(k - k_0)^2) \tag{3}$$

where τ is the bandwidth of the filter, which decides how wide or narrow the filter is, and k_0 is the center frequency that determines where the filter focuses on. This filter is applied simply by multiplying it with the data in the Fourier domain. Transforming the product back into the time domain afterward will remove the noise in the original data.

3 Algorithm Implementation and Development

The algorithms used in this project were for averaging the data, which gives us the time frequency, and filtering, which removes noise and aids in the acquisition of location data. The MATLAB code for both of these is available in **Appendix B**.
Before performing these algorithms, there were several steps to prepare the data. First, we divided x into 64 points to fit the FFT requirement of 2^n. Additionally, we scale k, the wave number, by $\frac{2\pi}{L}$ for the FFT, which as mentioned in section 2, runs on 2π periodic signals.
Additionally, the `reshape` command is used to change the `262144 x 49` data into a `64 x 64 x 64 x 49` array, which represents the frequency locations in 3-D space, for 49 different time steps. This allows the information in `subdata.mat` to be easily processed by our Fourier Transforms.

3.1 Signal Averaging

Algorithm 1: Signal Averaging Algorithm

Import data from `subdata.mat`
Initalize average as 0
for $j = 1 : 49$ **do**
  Reshape data at timestep j into matrix
  Perform Fourier Transform on data using `fftn` and add it to the average
end for
Use `fftshift` to shift average back from the FFT, then divide take the absolute value and divide by 49
Find the maximum index among the averaged signal using `ind2sub`, and then locate the center frequency

This algorithm allows us to average the signal data, and then obtain a center frequency.

3.2 Signal Filtering

This loop applies the filter centered on the frequency found from averaging to the collected data. Using the results, we can plot both the isosurface, which shows the area where the filtered signal is detected. We also store the exact location of the submarine at each time step. τ was determined to be 0.01 by finding a suitable value where each isosurface was just small enough to visualize a submarine location.

Algorithm 2: Signal Filtering Algorithm

Set $\tau = 0.01$
Define the Gaussian filter
for $j = 1 : 49$ **do**
  Reshape data at timestep j into matrix
  Perform Fast Fourier Transform on data using `fftn` and shift it to frequency using `fftshift`
  Multiply transformed frequency data by filter
  Plot isosurface where signal is detected
  Store location data
end for

4 Computational Results

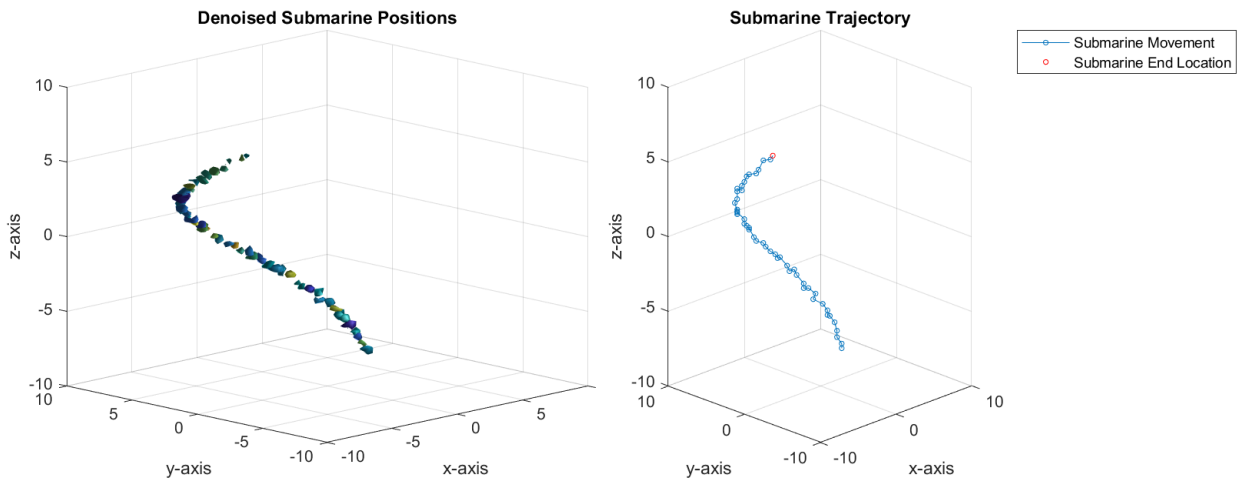
After performing signal averaging, we found a central frequency of $(5.3407, -6.9115, 2.1991)$. Using this in the Gaussian filter, we find that the final position of the submarine is $(-5.3125, 0.9375, 6.25)$. Thus we should send our aircraft to $(-5.3125, 0.9375)$. The rest of the positions at each time step are in the table below:

x	y	z
3.125	0.3125	-8.125
3.125	0.3125	-7.8125
3.125	0.9375	-7.5
3.4375	1.25	-7.1875
3.125	1.25	-6.5625
3.125	1.875	-6.25
3.125	2.1875	-6.25
3.125	2.1875	-5.9375
3.125	2.8125	-5.625
2.5	3.4375	-5.3125
2.8125	3.4375	-5.
2.5	4.0625	-4.6875
2.1875	4.375	-4.6875
2.1875	4.375	-4.37
1.5625	4.6875	-3.75
1.5625	5	-3.4375
0.9375	5	-3.4375
0.9375	5.3125	-3.125
0.3125	5.625	-2.5
0	5.625	-2.5
-0.3125	5.625	-2.1875
-0.9375	5.625	-1.875
-1.25	5.9375	-1.5625
-1.5625	5.9375	-1.25
-2.5	5.9375	-0.9375

x (cont.)	y (cont.)	z (cont.)
-2.8125	5.9375	-0.625
-3.125	6.25	0
-3.4375	5.9375	0
-3.75	5.9375	0.3125
-4.375	5.625	0.625
-4.375	5.625	0.9375
-5.3125	5.625	1.5625
-5.625	5.3125	1.5625
-5.625	5.3125	1.875
-5.9375	5	1.875
-6.25	5	2.5
-6.25	4.6875	2.8125
-6.5625	4.375	3.4375
-6.875	4.0625	3.75
-6.875	3.4375	3.75
-6.875	3.4375	4.0625
-6.875	3.125	4.375
-6.5625	3.125	4.6875
-6.875	2.5	5
-6.25	2.1875	5
-6.25	1.875	5.3125
-5.9375	1.5625	5.9375
-5.3125	1.25	5.9375
-5.3125	0.9375	6.25
x final	y final	z final

Table 1: Submarine position in all 49 time steps

Additionally, both the isosurfaces and line plot of the submarine trajectory are plotted below:



(a) Isosurface graph

(b) plot3 graph

Figure 1: (Left) 3-D Isosurface plot of denoised submarine positions, with $\tau = 0.01$ (Right) 3-D plot of submarine trajectory, with final location in red

5 Summary and Conclusions

In summary, we see that the by averaging frequency in the Fourier domain, using FFT, and applying a filter, we can obtain meaningful results from previously noisy data. We first found the central frequency of (5.3407, -6.9115, 2.1991). Next, we removed extraneous frequencies using a Gaussian Filter with $\tau = 0.01$. Finally, analyzing the denoised data reveals the submarine's trajectory and its final position of (-5.3125, 0.9375, 6.25), so we should send the aircraft to (-5.3125, 0.9375).

Appendix A MATLAB Functions

- `abs(X)`: Returns the absolute value of `X`.
- `exp(X)`: Returns the exponential function e^X for the inputted `X`.
- `fftn(U)`: Returns the n-dimensional Fast Fourier Transform of the data `U`.
- `fftshift(U)`: Returns a rearranged Fourier Transform by shifting the zero frequency component to the center of the array, given an input `U`.
- `ifftn(U)`: Returns the n-dimensional inverse Fast Fourier Transform given data `U`.
- `ind2sub(sz, ind)`: Returns the indices of the matrix which correspond to the indices defined by `ind` in a square matrix defined with size `sz`.
- `isosurface(X,Y,Z,U,value`: Given data `U`, plots the isovolume surface at coordinates defined by `X`, `Y`, and `Z`.
- `linspace(x,y,n)`: Returns n evenly spaced points between x and y.
- `max(X)`: Returns the maximum value in the inputted `X`.
- `meshgrid(x,y,z)`: Returns 3 dimensional grid coordinates as defined by x, y, and z
- `reshape(U,n,n,n)`: Returns rearranged data in a `n x n x n` sized matrix, given the data `U`.
- `zeros(i,j,k)`: Returns an `i x j x k` sized matrix with all entries as 0.

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);

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(0.5)
end
close all

%% Averaging Signal
Utn_ave=zeros(n,n,n);
for j=1:49
Un(:,:,:)=reshape(subdata(:,j),n,n,n);
Utn_ave=Utn_ave+fftn(Un);
end
Utn_ave=abs(fftshift(Utn_ave))/49;
[m, index] = max(Utn_ave(:)); %
[ii,jj,ll] = ind2sub(size(Utn_ave),index);
x=ks(jj); y=ks(ii); z=ks(ll);

%% Filter Data

tau=0.01;
filter=exp(-tau*(((Kx-x).^2)+((Ky-y).^2)+((Kz-z).^2))); % Define the
    filter

x_coord = zeros(1,49); y_coord = zeros(1,49); z_coord = zeros(1,49);

for j=1:49
    Un(:,:,:)=reshape(subdata(:,j),n,n,n);
    Un_shift = fftshift(fftn(Un));
    Utn_filter= Un_shift.*filter;
    Unf = ifftn(Utn_filter);

    %% Figure 1
    isosurface(X,Y,Z,abs(Unf),0.9*max(abs(Unf(:))))
    gca.FontSize = 14;
    view(-45,15)
```

```
    axis([-L L -L L -L L]), grid on, drawnow
    xlabel('x-axis'); ylabel('y-axis'); zlabel('z-axis')
    title("Denoised Submarine Positions")
    pause(0.05)

    %% Storing Submarine Location
    [maxx, index] = max(Unf(:));
    [ii,jj,ll] = ind2sub(size(Unf),index);
    x_coord(j) = X(ii,jj,ll); y_coord(j) = Y(ii,jj,ll); z_coord(j) = Z(ii,
        jj,ll);
end
print -depsc sub_location.eps
pause(2)

%% Figure 2

close all
plot3(x_coord,y_coord,z_coord, '-o', 'MarkerSize', 3)
hold on
plot3(x_coord(end),y_coord(end),z_coord(end), 'ro', 'MarkerSize', 3)
title('Submarine Trajectory')
legend('Submarine Movement','Submarine End Location')
xlabel('x-axis'); ylabel('y-axis'); zlabel('z-axis')
gca.FontSize = 14;
view(-45,15)
axis([-L L -L L -L L]), grid on, drawnow

print -depsc sub_final.eps
```