AMATH 482 Assignment 1:
Tracking a Submarine Using Acoustic Signature

Yuqing Cui

January 27, 2021

Abstract

In this report, we are provided with noisy acoustic data measured over a 24-hour period in half-hour increments to hunt for a moving submarine in the Puget Sound. We transferred the data into the frequency domain using the fast Fourier transform and determines the center frequency by averaging the spectrum. Next, we generated a Gaussian filter to denoise the data and transferred data back into the spacial domain using the inverse fast Fourier transform. We consequently obtained the path of the submarine and located its final position.

I. Introduction and Overview

Sound data in real world are usually associated with noises, which would significantly obstruct analyzing and seeking useful information. Luckily, we are able to reduce those noises with many mathematical theories. In this assignment, we are given noisy acoustic data of a moving submarine in complex values. Those data are measured in 49 time steps over a 24-hour period with a half-hour increments. We will use fast Fourier transform to shift our inputs into the frequency domain, normalizing and averaging them. The purpose of this report is to filter out noises with a Gaussian filter and to eventually capture the movement and the final position of the submarine.

II. Theoretical Background

The three main mathematical functions we applied in this report are fast Fourier transform (FFT), inverse fast Fourier transform (IFFT), and the Gaussian filter.

If we have a function $f(x)$ with $x \in \mathbb{R}$, the Fourier transform of $f(x)$, which is denoted by $\hat{f}(k)$, is defined by the formula

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

We can also apply the inverse Fourier transform to retrieve $f(x)$ when the corresponding $\hat{f}(k)$ is given

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

in which $e^{i\theta} = cos(\theta) + isin(\theta)$ by Euler's formula, $f(x)$ is a function of space or time, and $\hat{f}(k)$ is a function of frequencies k.

In this paper, we are applying fast Fourier transform with a complexity of $O(Nlog(N))$, which is much faster than discrete Fourier transform (DFT) with a complexity of $O(N^2)$

$$\hat{x}(k) = \frac{1}{N}\sum_{n=0}^{N-1} x_n e^{\frac{2\pi i k n}{N}}. \tag{3}$$

In order to perform FFT, we have to use the divide and conquer algorithm to split a DFT of length N into powers of 2 (i.e. 2^n).

The next essential mathematical theory we would implementing is the Gaussian filter. We reduce noises in our data by multiplying the signal in the frequency domain with the filter function. A simple 1-D Gaussian filter is defined as

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

where τ is the width of the filter that is manually determined and k_0 is the center frequency that we want to filter around. By applying the Gaussian filter, the center frequency would be amplified while others would be reduced. In this report, we are going to apply a 3-D filter

$$F\left(k_x, k_y, k_z\right) = e^{-\tau((k_x-k_{x_0})^2+(k_y-k_{y_0})^2+(k_z-k_{z_0})^2)}, \tag{5}$$

III. Algorithm Implementation and Development

We first created a 3-D structure of shape $64 \times 64 \times 64$ for further computation because the fast Fourier transform requires 2^n data points in order to perform well. Then we rescaled the frequency k by a factor of $\frac{2\pi}{L}$ because FFT assumes 2π periodic signals. We then set up a grid for later plotting.

Next, we reshaped our data into our built structure and used FFT to transfer those data into the frequency domain. We took the sum of the frequencies and averaged them by the total time steps 49. Since the position data are complex values, we took the absolute value to gain their complex magnitudes. By obtaining the maximum and the corresponding index using the `max` method, we were able to locate the coordinates of the center frequency with the `ind2sub` function. We finally mapped those coordinates back to our grid in order to get our true center frequency.

Lastly, with the calculated center frequency and a chosen parameter $\tau = 0.2$, we generate a Gaussian filter. We applied the filter around the center frequency to denoise our data. Then we used the inverse fast Fourier transform to transfer the processed data back into spacial domain. Similarly, we obtained the maximum and the corresponding index using the `max` method, and mapped those indices back to get the real spacial coordinates of the submarine.

IV. Computational Results

After averaging the spectrum, we determined the center frequency to be $(-4.7124, 3.1416, -7.8540)$. With the frequency, we are able to generate the Gaussian filter to denoise the data. We set $\tau = 0.2$ in our filter equation, and the path is shown in **Figure 1**. We have tried many values for τ and have found out that the range for τ that would produce clear output is wide: any value within the range $[0.05, 2]$ is acceptable. If τ is too small, there would still be many noises. If τ is too large, the path would be tortuous. In both cases, the trajectory would be unreadable. **Figure 2** shows the trajectory with a small τ value ($\tau = 0.02$) and **Figure 3** shows the trajectory with a large τ value ($\tau = 3$). We can see the submarine is moving upward with a circular path.

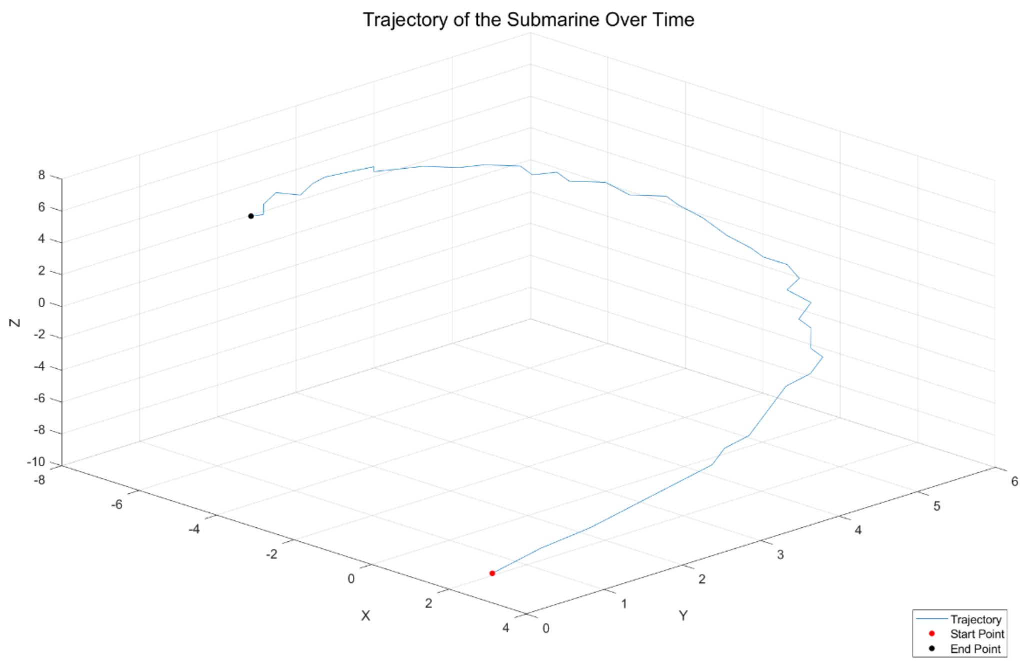

Figure 1. The trajectory of the submarine over time ($\tau = 0.2$)

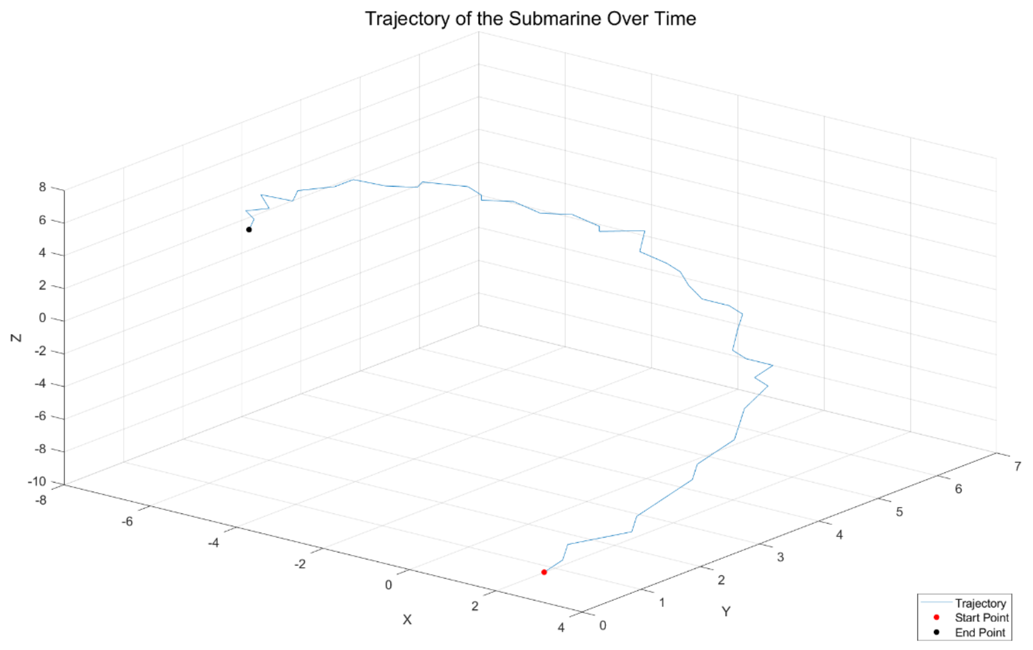

Figure 2. The trajectory of the submarine over time ($\tau = 0.02$)

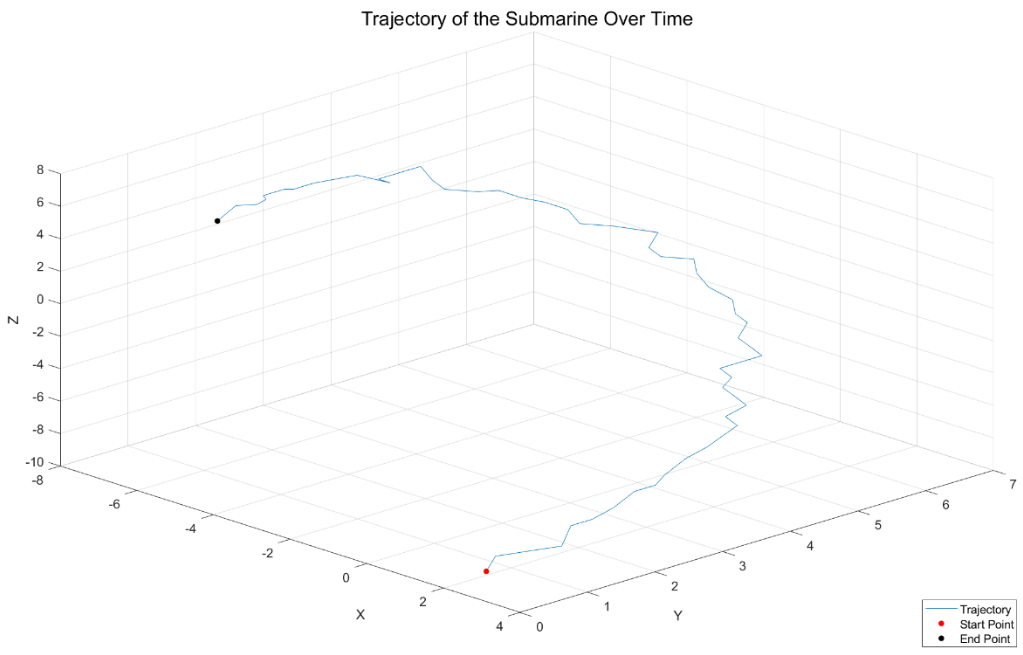

Figure 3. The trajectory of the submarine over time ($\tau = 3$)

We also obtained the x and y coordinates of the submarine at each step shown in **Table 1** below. The final position of the submarine, which is shown at $t = 49$, is at $(-5, 0.9375)$. Therefore, we should send our P-8 Poseidon subtracking aircraft to that position in order to catch the submarine. Times are measured over a 24-hour span at half-hour increments.

Table 1. The position of the submarine at each time

Time	1	2	3	4	5	6	7	8	9
x	3.1250	3.1250	3.1250	3.1250	3.1250	3.1250	3.1250	3.1250	3.1250
y	0	0.3125	0.6250	1.2500	1.5625	1.8750	2.1875	2.5000	2.8125

Time	10	11	12	13	14	15	16	17	18
x	2.8125	2.8125	2.5000	2.1875	1.8750	1.8750	1.5625	1.2500	0.6250
y	3.1250	3.4375	3.7500	4.0625	4.3750	4.6875	5	5	5.3125

Time	19	20	21	22	23	24	25	26	27
x	0.3125	0	-0.6250	-0.9375	-1.2500	-1.8750	-2.1875	-2.8125	-3.1250
y	5.3125	5.6250	5.6250	5.9375	5.9375	5.9375	5.9375	5.9375	5.9375

Time	28	29	30	31	32	33	34	35	36
x	-3.4375	-4.0625	-4.3750	-4.6875	-5.3125	-5.6250	-5.9375	-5.9375	-6.2500
y	5.9375	5.9375	5.9375	5.6250	5.6250	5.3125	5.3125	5	5

Time	37	38	39	40	41	42	43	44	45
x	-6.5625	-6.5625	-6.8750	-6.8750	-6.8750	-6.8750	-6.8750	-6.5625	-6.2500
y	4.6875	4.3750	4.0625	3.7500	3.4375	3.4375	2.8125	2.5000	2.1875

Time	46	47	48	49
x	-6.2500	-5.9375	-5.3125	-5
y	1.8750	1.5625	1.2500	0.9375

V. Summary and Conclusions

In conclusion, we successfully tracked the submarine with the given noisy data. We first transferred the data into frequency domain and averaged the spectrum to find the center frequency $(-4.7124, 3.1416, -7.8540)$ in order to apply a Gaussian filter at. Then we denoised the data using a filter with $\tau = 0.2$ and transferred it back into the spacial domain. We calculated the position of the submarine at each time and plotted its movement with those data points. We found out that the final position of the submarine is at $(-5, 0.9375)$, which is also the coordinates to send our P-8 Poseidon subtracking aircraft to.

Appendix A. MATLAB functions[1]

`load(filename)`: Loads data from the file with the given name. We used it to input data from subdata.mat.

`linspace(x1,x2,n)`: Generates `n` points with a even spacing between `x1` and `x2`. We used it to generate an evenly spaced vector for `meshgrid` later.

`fftshift(X)`: Rearranges a Fourier transform `X` by shifting the zero-frequency component to the center of the array. We used it to fix the visualization of the transformed data.

`[X,Y,Z]=meshgrid(x,y,z)`: Returns 3-D grid coordinates defined by the vectors `x`, `y`, and `z`. The grid represented by `X`, `Y`, and `Z` has size length(`y`)-by-length(`x`)-by-length(`z`). We used it to create axes for plotting later.

`zeros(n1,n2,n3)`: Returns an `n1`-by-`n2`-by-`n3` array of zeros. We used it to create new arrays to store outputs.

`reshape(A,sz)`: Reshapes array `A` using `sz` to define the size. We used it to turn our input data into the shape of `64*64*64` in order to apply fast Fourier transform later.

`abs(X)`: Returns the absolute value of each element in array `X`. Since `X` is complex here, we used it to return the complex magnitude.

`fftn(X)`: Returns the multidimensional Fourier transform of an N-D array `X` using a fast Fourier transform algorithm. We used it to transform our data into the frequency domain.

`[M,I] = max(A)`: Returns the maximum `M` and the corresponding index `I` of matrix `A`. We used it to gain the index in order to find the center frequency using `ind2sub`.

`[x,y,z] = ind2sub(sz,ind)`: Returns the coordinates `(x,y,z)` given an index `ind` of a matrix of size `sz`. We used it to find the coordinates of the center frequency.

`ifftn(X)`: Returns the multidimensional discrete inverse Fourier transform of an N-D array `X` using a fast Fourier transform algorithm. We used it to transfer our denoised data after applying the filter back into the spacial domain.

`plot3(x,y,z)`: Plots coordinates in 3-D space. We used it to plot the trajectory of the submarine.

`table(X,Y,'VariableNames',{'x','y'})`: Generates a table showing vector `X` and `Y` with variable names defined. We used it to visualize **Table 1**.

[1] Basic function descriptions from MathWorks Help Center.

Appendix B. MATLAB codes

```
%% Given 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(1)
% end

%% Part 1 Center Frequency
% average data
ave = zeros(n,n,n);
for j = 1:49
   utnave(:,:,:) = reshape(subdata(:,j),n,n,n);
   ave = ave + utnave;
end
ave = abs(fftn(ave))/49;

% center frequency
[utnmax,ind] = max(ave(:));
[cx,cy,cz] = ind2sub([64,64,64],ind);
centerFreq = [Kx(cx,cy,cz) Ky(cx,cy,cz) Kz(cx,cy,cz)];

%% Part 2 Plot Trajectory
% Gaussian filter
tau = 0.2;
filter = exp(-tau.*((Kx-centerFreq(1)).^2 +
        (Ky-centerFreq(2)).^2 + (Kz-centerFreq(3)).^2));
```

```
% get positions
for i = 1:49
    utn = fftn(reshape(subdata(:,i),n,n,n));
    unf = ifftn(filter .* utn);
    [unfmax,ind] = max(abs(unf(:)));
    [x,y,z] = ind2sub([64,64,64],ind);
    xpos(i) = X(x,y,z);
    ypos(i) = Y(x,y,z);
    zpos(i) = Z(x,y,z);
end

% plot trajectory
plot3(xpos,ypos,zpos)
set(gca,'Fontsize',12)
xlabel('X')
ylabel('Y')
zlabel('Z')
title('Trajectory of the Submarine Over Time','Fontsize',18)
grid on, hold on

% start point
plot3(xpos(1),ypos(1),zpos(1),'r.','Markersize',18)
% end point
plot3(xpos(49),ypos(49),zpos(49),'k.','Markersize',18)
legend('Trajectory','Start Point','End Point')

%% Part 3 Table of x and y position
Table = table(xpos', ypos', 'VariableNames', {'x', 'y'})
```