AMATH 482 Assignment 1

Carter Peyton

January 27, 2021

Abstract

This assignment aims to extract information from a noisy signal via averaging and filtering. The Fourier transform is used to change the signal into frequency space where it is then averaged to remove noise and find a central frequency. This central frequency corresponds to the frequency of a moving object, and the signal is then filtered around the central frequency. The resulting signal allows us to plot the trajectory of this object over the same course of time as our signal.

1 Introduction and Overview

Suppose that you are hunting for a submarine in the Puget Sound with some noisy, acoustic signal. The submarine emits an unknown frequency and moves over the course of your recording. Through averaging the signal in the frequency spectrum, the noise in the signal will lessen and leave only the frequency of the submarine left. Once the submarine's frequency signature is known, a Gaussian filter can be applied to the acoustic signal to further isolate the submarine's signal. The resulting signal can be converted back into the time domain, which will provide the necessary information to track the course of the submarine.

2 Theoretical Background

2.1 The Fourier transform

The Fourier transform is a transform applied to functions that decomposes them into their frequency components. Given some function, $f(x)$, the Fourier transform, $F(k)$ is

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

2.1.1 The inverse Fourier transform

There exists the inverse Fourier transform, which takes a function from the frequency domain back into the time domain. This will be useful for finding the location of the submarine with respect to time.

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

2.1.2 Fast Fourier transform (FFT)

The computational form of the Fourier transform is called the Fast Fourier transform (FFT). The FFT is an algorithm that does exactly what it says, it computes the Fourier transform of some function over an interval in a fast manner. However, the algorithm works over some discrete space and assumes that certain conditions exist. First, it assumes that it operates over an interval from -2π to 2π. Second, it assumes that the number of discrete points in this interval is some power of 2 (i.e $n = 2^i$ where n is the number of points).

2.2 Filtering

In order to pick out the signal of a certain frequency (called the central frequency) the original signal can be multiplied by some filtering function $g(t)$. The Gaussian in Cartesian space is

$$g(x, y, ...) = e^{-\tau*((x-x_0)^2+(y-y_0)^2+...)} \tag{3}$$

In frequency space this becomes

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

Where k_0 is the center frequency and τ controls the width of the Gaussian. A larger value of τ corresponds to a thinner Gaussian.

3 Algorithm Implementation and Development

3.1 Initialization

The following algorithm imports the acoustic data into MATLAB and sets up the Fourier domain. The domain goes from -10 to 10 in Cartesian space, and it needs to be scaled by $2\pi/L$ ($L = 10 - (-10)$) and contains 64 Fourier modes (2^6). Also notice that we need to use fftshift, which moves the second half of frequencies to the left of the first half (this is necessary because FFT swaps the two initially, so a second swap brings them back into the right order). See "Part 0" of the MATLAB code in Appendix B.

3.2 Averaging

There are three steps necessary to finding the central frequency (that of the submarine): remove noise, find the coordinates in frequency space that correspond to the maximum frequency, and use those coordinates to find the submarine's frequency. We assume the noise in our signal is white noise, which means that it will sum to 0 over the course of the signal. This property is what allows averaging to "pick" out signals. See "Part 1" of the MATLAB code in Appendix B.

3.2.1 Removing noise

The submarine data is given in a 262144x49 matrix, this corresponds to 49 instances in time of a 64x64x64 matrix containing acoustic data in Cartesian space. First we loop through each instance of time and convert the 262144 entries into a 64x64x64 matrix (called Un). We then perform the FFT on Un and add the resulting matrix to our average matrix (at this point the matrix is only a sum). To get the true average matrix, we perform a fftshift (remember that FFT swaps the first and second halves of frequency), take the absolute value, and divide it by our 49 (the number of times that we just looped through).

3.2.2 Finding maximum frequency coordinates

For simplicity, we can convert our 64x64x64 average matrix into a 262144x1 vector. This allows us to easily find the index at which the maximum frequency occurs. Using the MATLAB command ind2sub, we can convert the index from the 1 dimensional vector into the 3 dimensional coordinates for that frequency.

3.2.3 Converting to frequency

Importantly, these coordinates are not the frequency itself. To find the maximum frequencies in each dimension we need to plug these coordinates into our Fourier space ($k_{x0} = K_x(i, j, k)$ corresponds to the central frequency in x).

3.3 Filtering

See "Part 2" of the MATLAB code in Appendix B.

3.3.1 Creating and applying a 3D Gaussian

Once we have obtained the frequency values to filter around, we need to construct a 3D Gaussian filter. Once the filter is constructed, we loop through each instance of sub data time, apply the Fourier transform, multiply the resulting function by the filter function, and transform that back into the time domain.

3.3.2 Finding the coordinates

We can apply a similar technique to finding the maximum frequencies to our filtered data (see section 3.2.2). The key difference is that we obtain indices for all 49 values in time. These indices can then be used in our x, y, and z vectors respectively to find the x, y, z coordinates of the submarine at each point in time.

3.3.3 Plotting the path of the submarine

Using the MATLAB command plot3 we can plot the x, y, and z values for all 49 points in time.

3.4 Reporting x and y coordinates

In order to list the x and y coordinates that the P-8 Poseidon should go to, we can generate a table that contains our x and y position vectors. See "Part 3" of the MATLAB code in Appendix B.

4 Computational Results

4.1 Center Frequency

The following central frequencies are obtained for the x, y, and z dimensions respectively $k_x = 5.34, k_y = -6.91, k_z = 2.19$.

4.2 Submarine Trajectory

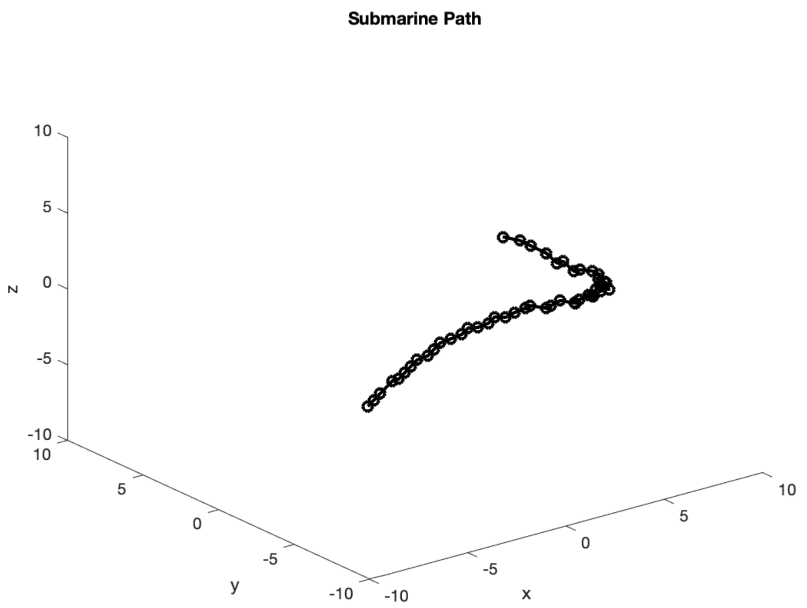
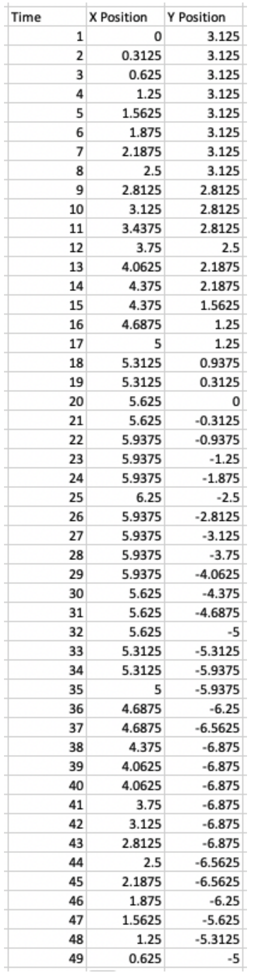


Figure 1: Path of the submarine over time (filter window size of 2). The submarine starts at the bottom and travels upward along the trajectory.

4.3 X and Y Coordinates



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

Figure 2: This table contains the x and y coordinates the P-8 Poseidon should use in order to track the submarine

Ultimately the P-8 Poseidon aircraft should be sent to the coordinates $(0.625, -5.00)$ in order to destroy the submarine.

5 Summary and Conclusions

The central frequencies of the submarine are $k_x = 5.34, k_y = -6.91, k_z = 2.19$. The final coordinates of the submarine are $(0.625, -5.00, 6.56)$, and the P-8 Poseidon aircraft should go to $(0.625, -5.00)$ in order to destroy the submarine. Through the Fourier transform, averaging, and filtering we extracted relevant data from an otherwise noisy signal.

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.

- `kx = fftshift(k)` rearranges a Fourier transform X by shifting the zero-frequency component to the center of the array.
- `[X,Y,Z] = meshgrid(x,y,z)` returns 2-D grid coordinates based on the coordinates contained in 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.
- `avg = zeros(n,n,n)` creates an n x n x n matrix of zeros.
- `B = reshape(A,sz)` reshape(A,sz) reshapes A using the size vector, sz, to define size(B).
- `Y = fftn(X)` returns the multidimensional Fourier transform of an N-D array using a fast Fourier transform algorithm.
- `[val,ind] = max(oneD)` returns the maximum value and corresponding index of the vector oneD
- `[i,j,k] = ind2sub(s,ind)` returns 3 subscript arrays i,j, and k containing the equivalent multidimensional array subscripts equivalent to ind for an array of size s.
- `I = ifftn(F)` computes the inverse multidimensional fast Fourier transform.

Appendix B MATLAB Code

```matlab
%% Part 0 - Initialize
% 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);


%% Part 1 - average
avg = zeros(n,n,n);
for j=1:49
    Un(:,:,:)=reshape(subdata(:,j),n,n,n);
    unf = fftn(Un);
    avg = avg + unf;
end
avg = abs(fftshift(avg))/49;
oneD = avg(:);
% Find the center frequency
[val,ind] = max(oneD)
s = [64,64,64];
[i,j,k] = ind2sub(s,ind) % prints indices of the max
% Central frequencies
kx_center = Kx(i,j,k)
ky_center = Ky(i,j,k)
kz_center = Kz(i,j,k)
```

```
%% Part 2 - filter around the central freq
a = 2; % window size
filter = exp(-a*((Kx-kx_center).^2 + (Ky-ky_center).^2 + (Kz-kz_center).^2));

for index = 1:49
    Un(:,:,:)=reshape(subdata(:,index),n,n,n);
    unf = fftn(Un);
    un_filt = fftshift(unf).*filter;
    un = abs(ifftn(fftshift(un_filt))); % 64 x 64 x 64 (remove n for good curve)
    un_oneD = un(:);
    [val,ind] = max(un_oneD);
    [i(index),j(index),k(index)] = ind2sub(s,ind); % max indices for 1 time?
end
in = [i', j', k']; % gives a 49 x 3 matrix (49 times values in rows and indices in columns)

final_coords = [x(in(end,1)), y(in(end,2)), z(in(end,3))] % prints the final x, y, z coords

% Plot the path of submarine
plot3(x(in(:,1)),y(in(:,2)),z(in(:,3)),'k-o','LineWidth',2)
xlabel('x')
ylabel('y')
zlabel('z')
axis([-L L -L L -L L])
title('Submarine Path')
print('sub_path.png','-dpng')


%% Part 3 - generate x and y coordinate table
pos(:,1) = x(in(:,1));
pos(:,2) = y(in(:,2));
pos = table(pos);
writetable(pos,'positions.csv')
```

Code from homework1.m