John Curtis

Assignment 1: A Submarine Problem

Abstract

This paper explores the methods to track a submarine using noisy acoustic data. Fourier analysis and filtering were used to find the signature frequency of the submarine and denoise the data. The algorithm was written in MatLab making use of the Fast Fourier Transform. Through these methods, the path of the submarine was found with reasonable accuracy and its final (x, y) coordinates were found to be $(-5, 0.9375)$.

Introduction and Overview

In the real world, denoising data is a common problem as most if not all measured data is not perfect and has some noise and imperfections in it. In this scenario we have acoustic data for the Puget Sound area and are trying to locate a submarine using the acoustic date. The data contains the X, Y, and Z acoustic measurements for 49 measurements that occurred every half an hour for 24 hours. However, the data is very noisy so to get a clearer picture of the location of the submarine we must first denoise it. We will use Fourier analysis and filtering to find the acoustic signature of the submarine and reduce the influence of acoustic data other than that from the submarine. Using the denoised data we aim to be able to determine the location of the submarine under the water.

Theoretical Background

The Fourier transform can be used to decompose a function into the fundamental frequencies of sine and cosine functions that sum to make up the initial function. The discrete Fourier transform does a similar thing but instead of a continuous function, it finds the frequencies making up a discrete set of points. The inverse Fourier transform takes those frequencies and reproduces the data from them. By multiplying a filter function to the Fourier transform of a set of data, we can isolate the frequencies to a set range, one such filter being a Gaussian function. Therefore, if we have noisy data but know what frequency the data we are looking for is in, then we can apply the Fourier transform, multiply the filter function and then apply the inverse Fourier transform, we can remove or at least reduce all values that are not in the filtered frequency range, so we can more clearly see the data at the frequency we want to observe.

However, we do not always know what frequency we are looking for which is where the idea of averaging comes in. White noise which is a common type of noise in measurements affects all frequencies equally with a mean of 0, thus if we average the frequencies of the signal over different realisations then the noise will approach the mean of 0 and be cancelled out, while the non-noise signal we are looking for will remain. This way we can isolate what frequency the non-noise data is at and use a filter function around that frequency to remove the noise from the initial data. We also do not need to worry about what time or position the certain frequencies occur when averaging since a property of the Fourier transform is that the frequencies will all be aligned to the centre after the Fourier transform, even if they occur at different times; in other words, the Fourier transform ignores the time information and only extracts the frequency information of the signal.

Algorithm Implementation and Development

The algorithm to denoise the submarine data was coded using MatLab.

Firstly, I created a set of frequencies scaled by $\frac{2\pi}{L}$ since MatLab's Fourier transform function assumes that the input has a period of 2π. I also created a set of position values, and then split both the frequency and position values into 3 separate matrices that represent the x, y and z planes.

The submarine data contains 49 realisations of the acoustic data in 3-dimensions with 64 recorded values along each dimension. The initial data was in the form of a 262144×49 matrix so I reshaped it to a 49 different $64 \times 64 \times 64$ matrices, where each matrix represents a single time realisation of the acoustic data and each dimension of the matrix represents the 3 dimensions of space.

Then for each matrix I applied the 3-D Fourier transform and added all of the 49 Fourier transforms of the data. After that I took the absolute value of the sum to get the real magnitude of the values rather than imaginary, and then divided by 49 to get the average signal in frequency space. As explained earlier, averaging the frequencies should have reduced the noise of the data overall allowing us to easily find the signature frequency of the submarine. I took the maximum value of the averaged frequency data and more importantly the matrix indices at which the maximum occurred. I used these indices to find the corresponding frequencies in the frequency matrices I had set up earlier to find the signature frequency of the submarine in the x, y and z directions separately.

I created a filter using the Gaussian function with the following formula:

$$e^{-a(K-K_0)^2} \qquad Eq.1$$

Where K are the frequencies and K_0 is the submarine signature frequency, in each of the x, y and z directions. The variable a is inversely proportional to the width of the filter and through trial and error the optimum value was found to be $a = 0.5$. Then by multiplying the 3 1-D filter functions together, I created a single 3-D filter function.

I then multiplied this filter function to the each of the 49 Fourier transformed measurements to reduce all frequencies except for the submarine's frequency. After which I took the inverse 3-D Fourier transform of them, and then the absolute to get the data as real values. Then for each realisation, I found the x, y and z coordinate at which the maximum value occurs to get the position of the submarine at each of the 49 time increments.

Computational Results

Using the described algorithm, the signature frequencies for the submarine along each axis were found to be:

- $Kx = 5.3407$
- $Kx = -6.9115$
- $Kx = 2.1991$

Using these frequencies as the centre of the filter to denoise the data, the path of the submarine was found. The following plot and table show the x, y and z values of the position at each of the 49 30-minute time intervals:

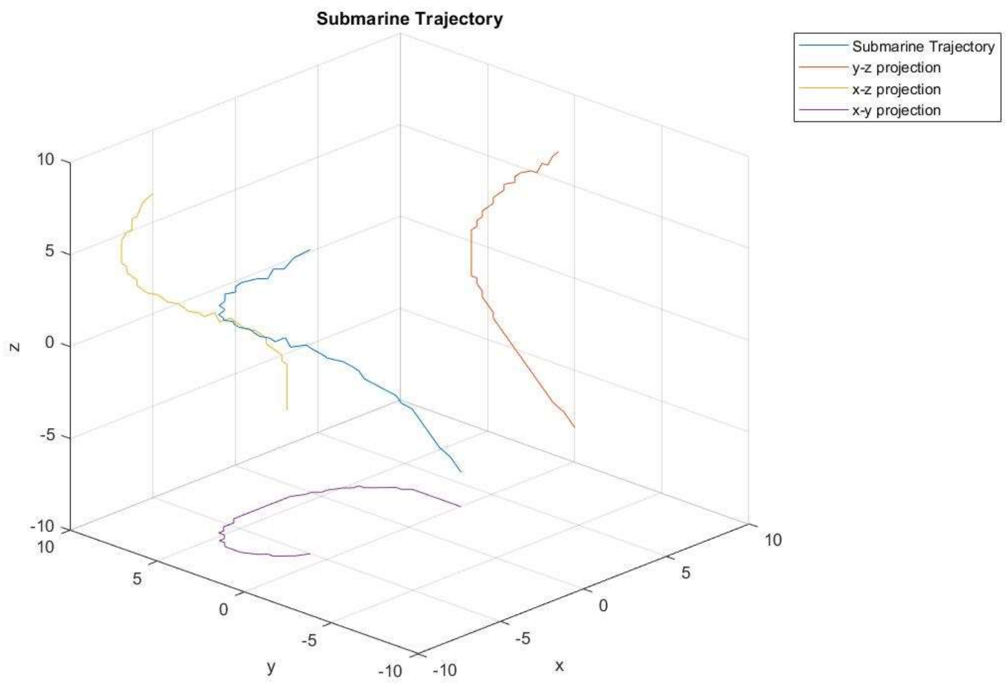

Figure 1 - Submarine Trajectory

Time	1	2	3	4	5	6	7	8	9	10
x	3.1250	3.1250	3.1250	3.1250	3.1250	3.1250	3.1250	3.1250	3.1250	2.8125
y	0.0000	0.3125	0.6250	1.2500	1.5625	1.8750	2.1875	2.5000	2.8125	3.1250
z	-8.1250	-7.8125	-7.5000	-7.1875	-6.8750	-6.5625	-6.2500	-5.9375	-5.6250	-5.3125
Time	11	12	13	14	15	16	17	18	19	20
x	2.8125	2.5000	2.1875	1.8750	1.8750	1.5625	1.2500	0.6250	0.3125	0.0000
y	3.4375	3.7500	4.0625	4.3750	4.6875	4.6875	5.0000	5.3125	5.3125	5.6250
z	-5.0000	-4.6875	-4.3750	-4.0625	-3.7500	-3.4375	-3.1250	-2.8125	-2.5000	-2.1875
Time	21	22	23	24	25	26	27	28	29	30
x	-0.3125	-0.9375	-1.2500	-1.8750	-2.1875	-2.8125	-3.1250	-3.4375	-4.0625	-4.3750
y	5.6250	5.9375	5.9375	5.9375	5.9375	5.9375	5.9375	5.9375	5.9375	5.9375
z	-1.8750	-1.8750	-1.2500	-1.2500	-0.9375	-0.6250	-0.3125	0.0000	0.3125	0.6250
Time	31	32	33	34	35	36	37	38	39	40
x	-4.6875	-5.3125	-5.6250	-5.9375	-5.9375	-6.2500	-6.5625	-6.5625	-6.8750	-6.8750
y	5.6250	5.6250	5.3125	5.3125	5.0000	4.6875	4.6875	4.3750	4.0625	4.0625
z	0.9375	1.2500	1.5625	1.8750	2.1875	2.5000	2.8125	3.1250	3.4375	3.7500
Time	41	42	43	44	45	46	47	48	49	
x	-6.8750	-6.8750	-6.8750	-6.5625	-6.2500	-6.2500	-5.9375	-5.6250	-5.0000	
y	3.4375	3.4375	3.1250	2.5000	2.1875	1.8750	1.5625	1.2500	0.9375	
z	4.0625	4.3750	4.6875	5.0000	5.0000	5.6250	5.6250	6.2500	6.5625	

Figure 2 - Table of Coordinates of Submarine

Summary and Conclusion

From the results in the table from figure 2, the P-8 Poseidon submarine tracking aircraft should be sent to the (x, y) coordinates $(-5, 0.9375)$ as that is the latest position of the submarine.

Overall, we can see that using Fourier analysis to find the signature frequency and using frequency filtering to denoise data is effective and make it much easier to find useful information in the measured data. However it should be noted that this method did not remove all noise from the data as can be seen by some of the sharp corners and sudden jumps in figure 1. Though it is impractical to remove all noise altogether so for the purpose of denoising the data to track the submarine, the methods described in this paper did provide relatively accurate results.

Appendix A: MatLab Functions Used

- **Y = linspace(x1,x2,n)** – Generates n points between x1 and x2.
    - https://au.mathworks.com/help/matlab/ref/linspace.html
- **Y = meshgrid(x,y,z)** – Returns 3-D grid coordinates defined by the vectors x, y, and z.
    - https://au.mathworks.com/help/matlab/ref/meshgrid.html
- **Y = cell(n)** – Returns an n-by-n cell array of empty matrices.
    - https://au.mathworks.com/help/matlab/ref/cell.html
- **Y = zeros(sz1,...,szN)** – Returns an sz1-by-...-by-szN array of zeros.
    - https://au.mathworks.com/help/matlab/ref/zeros.html
- **Y = reshape(A,sz1,...,szN)** – Reshapes the array A into a sz1-by-...-by-szN array.
    - https://au.mathworks.com/help/matlab/ref/reshape.html
- **Y = fftn(X)** – returns the multidimensional Fourier transform of an N-D array using a fast Fourier transform algorithm.
    - https://au.mathworks.com/help/matlab/ref/fftn.html
- **Y = fftshift(X)** – rearranges a Fourier transform X by shifting the zero-frequency component to the center of the array.
    - https://au.mathworks.com/help/matlab/ref/fftshift.html
- **Y = abs(X)** – Returns the absolute value of each element in array X. If X is complex then instead returns the complex magnitude.
    - https://au.mathworks.com/help/matlab/ref/abs.html
- **[M,I] = max(A,[],’all’,'linear')** – Returns the maximum value in A as well as the linear index into A that corresponds to the maximum value in A.
    - https://au.mathworks.com/help/matlab/ref/max.html
- **[I1,I2,...,In] = ind2sub(sz,ind)** – Returns the equivalent multidimensional subscripts corresponding to the linear indices ind for a multidimensional array of size sz.
    - https://au.mathworks.com/help/matlab/ref/ind2sub.html
- **Y = exp(X)** – Returns the exponential e^x for each element in array X.
    - https://au.mathworks.com/help/matlab/ref/exp.html
- **Y = ifftn(X)** – Returns the multidimensional discrete inverse Fourier transform of an N-D array using a fast Fourier transform algorithm.
    - https://au.mathworks.com/help/matlab/ref/ifftn.html
- **writematrix(A,filename)** – Writes the matrix A to a csv file with the given filename.
    - https://au.mathworks.com/help/matlab/ref/writematrix.html
- **plot3(X,Y,Z)** – Plots coordinates in 3-D space.
    - https://au.mathworks.com/help/matlab/ref/plot3.html

Appendix B: MatLab Code

```
clear; close all; clc;

load subdata.mat;
L = 10;
n = 64;
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);

Rt = cell(1,49);
ave = zeros(n,n,n);
for j=1:49
    Un(:,:,:) = reshape(subdata(:,j),n,n,n);
    Unt = fftshift(fftn(Un));
    ave = ave + Unt;
    Rt{j} = Unt;
end
ave = abs(ave)/49;
[M,I] = max(ave,[],'all','linear');
[Ix,Iy,Iz] = ind2sub(size(ave),I);
Kx(Ix,Iy,Iz)
Ky(Ix,Iy,Iz)
Kz(Ix,Iy,Iz)

a = 0.5;
filterx = exp(-a.*(Kx-Kx(Ix,Iy,Iz)).^2);
filtery = exp(-a.*(Ky-Ky(Ix,Iy,Iz)).^2);
filterz = exp(-a.*(Kz-Kz(Ix,Iy,Iz)).^2);
filter = filterx.*filtery.*filterz;

locs = zeros(3,49);
for j=1:49
    Rfn = abs(ifftn(Rt{j}.*filter));
    [M,I] = max(Rfn,[],'all','linear');
    [Ix,Iy,Iz] = ind2sub(size(Rfn),I);
    locs(:,j) = [X(Ix,Iy,Iz),Y(Ix,Iy,Iz),Z(Ix,Iy,Iz)];
end
```

```
writematrix(locs,'locations.csv')
plot3(locs(1,:),locs(2,:),locs(3,:));
xlim([-10,10]); ylim([-10,10]); zlim([-10,10]);
xlabel("x"); ylabel("y"); zlabel("z");
grid on;
hold on;
plot3(10*ones(size(locs(1,:))),locs(2,:),locs(3,:));
plot3(locs(1,:),10*ones(size(locs(2,:))),locs(3,:));
plot3(locs(1,:),locs(2,:),-10*ones(size(locs(3,:))));
legend("Submarine Trajectory", "y-z projection", "x-z
projection", "x-y projection");
title("Submarine Trajectory")
```