AMATH 482 Homework 1

Alex Omusoru

January 24, 2020

Abstract

Given noisy acoustic data representing submarine movement, this paper aims to show how the signal was processed to determine the location of the submarine over the course of 24 hours. The data was transformed using a discrete Fast Fourier Transform and then filtered using a simple gaussian filter to allow for the extraction of the coordinates of the submarine. The path of the submarine was identified, and as a result the tracking aircraft was able to follow the submarine.

1 Introduction and Overview

The goal of this paper is to take noisy acoustic data of a segment of the Puget Sound and manipulate it in order to locate the submarine. The data was recorded over the course of 24 hours, in 30 minute increments (leading to 49 measurments). In order to determine the path, first the data was transformed using a discrete Fast Fourier Transform and then averaged over the time period. Then the central frequency was determined and identified as being the submarines unique acoustic signature. Once this was found, the noisy acoustic data was filtered based on the frequency in a simple gaussian filter, and then the coordinates of the maximum values were extracted and noted as the path of the submarine. Finally, the x and y coordinates of the submarine path were pulled out and used for the placement of a P-8 Poseidon subtracking aircraft which followed the submarine.

2 Theoretical Background

The discrete Fourier Transform (DFT) allows for the application of the Fourier Transform to discrete and finite data, in order to measure the composition of a signal broken down into frequencies. Given N real numbers $x_0, \ldots, x_{N-1}$, the DFT of the data is represented as follows:

$$\hat{x}_k = \frac{1}{N} \sum_{n=0}^{N-1} x_j e^{-ijk2\pi/N}, \quad k = -N/2, -N/2+1, \ldots, -1, 0, 1, 2, \ldots, N/2-1. \tag{1}$$

To convert back to the data, the inverse of the discrete Fourier Transform was used.

An algorithm called the Fast Fourier Transform (FFT) was later developed to speed up the computation, however it serves the same purpose as the DFT. The methods used in this paper utilize the FFT. In this paper the FFT was used to convert the data to a frequency domain, and then the Inverse Fast Fourier Transform (IFFT) was used to convert back to the space domain.

This paper utilizes a 3 dimensional simple gaussian filter. Based on the function

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

where τ represents the width of the filter and k_0 represents the center frequency to filter around.

In the paper, this formula is applied in a 3-dimensional extension. As a result the filter takes the form:

$$F(k_x, k_y, k_z) = e^{-\tau(k-k_{x0})^2} * e^{-\tau(k-k_{y0})^2} * e^{-\tau(k-k_{z0})^2} \tag{3}$$

The width of the filter is set to $\tau = 0.1$, while each k_0 represents the central frequency of the corresponding dimension.

3 Algorithm Implementation and Development

Algorithm **??** was used to compute the central frequencies. It assumes an established spacial domain, and n Fourier modes. It also assumes that the data from `subdata.m` is imported.
Algorithm **??** took the central frequency values in the k domain and was used to filter the data and compute the coordinates of the submarine in the space domain.

Algorithm 1: Central Frequency Algorithm

`UtnAvg` = n-by-n-by-n zero matrix
for $j = 1 : 49$ **do**
  Extract measurement j from `subdata`
  Apply FFT
  Add it to `UtnAvg`
end for
`UtnAvg =` absolute value of `UtnAvg` divided by 49
Determine the x y and z coordinates of the maximum value in `UtnAvg`
Multiply each by $2 * \pi/2 * L$ to prepare for filtration use

Algorithm 2: Filtration and Coordinate Location

define `tau` and create simple gaussian filter based on central frequencies
`coords` = 49-by-3 zero matrix for submarine coordinate cataloging
for $j = 1 : 49$ **do**
  Extract measurement j from `subdata` and apply the FFT
  Multiply by filter function
  Add it to `UtnAvg`
  Apply IFFT
  Determine the x y and z coordinates of the maximum value and add to `coords` on row j
end for

Algorithm 3: Filtration and Coordinate Location

`tau` = 0:step:T where T is the length of the song
`a` = width parameter
Initialize $ygt_specforstoringspectographdata$`for j = 1:length of` τ `do`
  `Build Gabor Filter using` τ `and a`
  `Multiply signal by filter function`
  `Apply FFT`
`Add it to` ygt_spec
`end for`
`Form spectograph using` $ygt_specasthedata.$

4 Computational Results

The central frequencies were determined to be $k_x = 3.4558, k_y = 15.7080, k_z = 12.5664$. Figure **??** represents the path of the submarine. The following are the (x,y) coordinates for the P-8 Poseidon subtracking aircraft to follow:

x	y
1	62

44 21
42 7
41 9
7 2
47 9
56 18
4 20
64 38
39 53
41 54
5 5
14 3
59 5
45 24
50 48
12 45
25 33
63 12
47 43
20 14
11 45
54 16
47 21
28 20
61 51
54 39
30 36
46 4
26 35
41 17
23 28
60 13
53 36
64 57
17 2
25 43
12 18
57 48
58 42
13 12
62 54
44 43
63 35
49 47
23 51
41 2
39 62
61 4

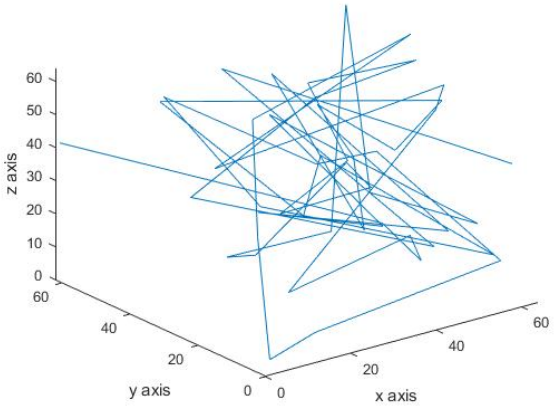

Figure 1: The path of the submarine plotted in 3-D.

5 Summary and Conclusions

Using Fourier Transform methods for discrete data and simple gaussian signal filters, the noisy acoustic data of the submarine movements were denoised and the path of the submarine was extracted. The x and y coordinates of the data were used for the P-8 Poseidon subtracking aircraft.

Appendix A MATLAB Functions

- `Y = fft(X)` computes the discrete Fourier transform (DFT) of X using a fast Fourier transform (FFT) algorithm.
- `X = ifft(Y)` computes the inverse discrete Fourier transform of Y using a fast Fourier transform algorithm. X is the same size as Y.
- `M = max(A,[],'all')` finds the maximum over all elements of A.
- `[row,col] = ind2sub(sz,ind)` returns the arrays row and col containing the equivalent row and column subscripts corresponding to the linear indices ind for a matrix of size sz. Here sz is a vector with two elements, where sz(1) specifies the number of rows and sz(2) specifies the number of columns.

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);

%% average fourier transform and determine central frequency
UtnAvg = zeros(n,n,n);
for j=1:49
    Un = reshape(subdata(:,j),n,n,n);
    Utn = fftn(Un);
    UtnAvg = UtnAvg + Utn;
end

UtnAvg = abs(fftshift(UtnAvg))/49;

M = max(UtnAvg,[],'all');
[Mx,My,Mz] = ind2sub(size(UtnAvg),find(abs(UtnAvg)== M))

Mx = (2*pi/(2*L))*Mx
My = (2*pi/(2*L))*My
Mz = (2*pi/(2*L))*Mz

%% filter original Un's and then shift back to space domain
tau = 0.1;
filter = exp(-tau*(Kx - Mx).^2).*exp(-tau*(Ky - My).^2).*exp(-tau*(Kz - Mz).^2);

coords = zeros(49, 3);
for j = 1:49
    Utn = fftshift(fftn(reshape(subdata(:,j),n,n,n)));
    Utnf = Utn.*filter;

    Unf = ifftn(ifftshift(Utnf));
    M = max(abs(Unf),[],'all');
    [coords(j, 1), coords(j, 2), coords(j, 3)] = ind2sub(size(Unf),find(abs(Unf)== M));
end


plot3(coords(:, 1), coords(:, 2), coords(:, 3))
xlabel("x axis")
ylabel("y axis")
zlabel("z axis")
```

Listing 1: Code used for Homework 1