AMATH 482 Homework 1

Yichen Shen

January 24, 2020

Abstract

This paper introduces the theoretical background, the algorithm implementation and development, and the computational results of the research on the problem of tracking a submarine using the noisy acoustic data that emits in the Puget Sound collected over a 24-hour period. This paper is going to implement averaging and filtering methods to the data in order to clean the spectrum and track the submarine using a sub-tracking aircraft.

1 Introduction and Overview

1.1 Background and Context

Giving a list of noisy acoustic data emits for a submarine, the objective is to utilize these data to hunt for a submarine in the Puget Sound. The submarine emits the mysterious acoustic frequency and these data are collected using a broad spectrum recording of acoustics over a 24-hour period in half-hour increments. Since the submarine is moving, one needs to determine its location and path using the propitiate mathematical tools, which will be address in the content.

1.2 Topic Importance

This is an important topic since, in real world, data always encountered noise, to locate the desire data one needs to perform a cleanup on the data. It is also important to understand the mathematical tool used in the cleanup process in order to achieve the objectives.

1.3 Objectives of Research

1. Determining the frequency signature generated by the submarine through averaging the spectrum.
2. Filtering the data around the central frequency in order to denoise the data and track the path of the submarine.
3. Determining the location of the submarine to send a P-8 Poseidon sub-tracking aircraft to track the location of the submarine.

2 Theoretical Background

2.1 Fourier Transform

As learned from the textbook [1], the Fourier transform converts a function of space (or time), x, into a frequency function of k.

The Fourier transform of f(x) is written as $\hat{f}(k)$, and its formula can be written as:

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

Furthermore, if wants to recover the space (or time) function fx from the given $\hat{f}k$, uses *inverse Fourier transform*, which formula is shown below:

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

2.1.1 Discrete Fourier Transform and Fast Fourier Transform

Known the continuous Fourier transform from above section, now consider the discrete function. When giving a set of discrete points and not knowing the high frequency phenomena, Discrete Fourier Transform (DFT) allows one to use a sequence (or vector) of N values sampled at equally-spaced points $\{ x_0, x_1, x_2, ..., x_{N-1}\}$, its sequence of numbers can be written as:

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

In this sequence of number, only finite frequencies given by k=0,1,2,...,N-1, are present.

Fast Fourier Transform is basically the same as the DFT but it perform DFT in a faster algorithm. When given a sequence of length N, FFT splits the DFT into two equally DFTs and repeat the process continuously. If DFT takes N operations to calculate N numbers its complexity is $O(N^2)$, FFT in this case, its complexity is $O(Nlog(N))$. When needs to process a large number of N, the FFT's operation is much more efficiency.

2.2 Filter with Gaussian function

When receiving a signal with noise, it is hard to detect where the pulse is coming from. Therefore, one needs to filter the noise in order to better analyze the signal. The filter that can be use is a function and then multiply it with the signal in the frequency domain. One simply choice of the filter functions is the Gaussian function:

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

τ is the width of the filter and constant k_0 is the centre of the filter in the frequency domain. The smaller the width the more signal it filter as the Gaussian decay is slower.

3 Algorithm Implementation and Development

3.1 Processing data into appropriate format

To begin, first load the `subdata.mat` into the work-space, sets the spatial domain to transform to be L = 10, and the Fourier modes to be n = 64 = 2^6. After that, discretize the domain on x, y, and z axis and select them periodically by using the first n points. Then re-scale the frequencies components by $2\pi/L$ since the FFT assumes 2π periodic signals.

To display the vectors in coordinate format with a size of length(j)-by-length(i)-by-length(k), uses meshgrid to return a 3D grid coordinates with {x,y,z} and {Kx,Ky,Kz}.

3.2 Averaging the spectrum

Find the central frequency of the signal generated by the submarine by averaging the spectrum. This can be done by sum up all of the signal then divide the sum by the total number of signal. Perform algorithm 1, which arithmetic are:

1. Define a summation variable with a 3D dimension using a n,n,n dimension zero array to collect all of the data.

2. Write a loop to repeat the process of modify the data for n times.
3. In the loop: first reshape the data on the columns to the n,n,n dimension.
4. In the loop: find the maximum element over all elements of the reshape data
5. In the loop: perform FFT to the reshaped 3D data using `fftn`, since the data are reshape into 3D dimension.
6. In the loop: sum the transferred data into the summation variable.
7. Outside the loop: find the average by dividing the absolute value of central shifted frequency (using `fftshift`) by 49.
8. Last, compute the isosurface data from the absolute value of average frequency, with an isovalue of 0.7 which impose a contour value of 0.7 to the data. Then find the coordinates of central frequencies using `mean()` accordingly.

Algorithm 1: Compute averaged transferred noisy signal

Import data from `subdata.mat`
Define `sum_ave` to be a zero array with $n * n * n$ dimension.
for $j = 1 : 49$ **do**
  Reshape data on the jth columns to n*n*n dimension
  Perform FFT by using `fftn` function to the reshaped 3D data
  Sum the transferred data into the `sum_ave`
end for
Find average value `ave` by dividing the absolute value of `fftshift(sum_ave)` by 49
Find the highest frequency of occurrence and normalize the signal
Compute isosurface data from absolute value of `ave` using `[f,v] = isosurface`
Extract average frequency's axes components from the isosurface data using `mean()`

3.3 Filtering around the central frequency

To apply the filter, using the Gaussian filter which can be referred to equation (4). In order to operate it in a 3D dimension, the filter will need to apply on each dimension, then multiply each filter together to get the filter to apply. Set the τ value to be 0.2 which is the width for the filter and the signal would damp around that width. The centres of the filter k_0 in three dimension is determined by the 3 coordinates value from the center frequency results in the previous section. Perform the algorithm 2 which arithmetic are:

1. Create a variable `loci` with a dimension of 49*3.
2. Write a loop to repeat the process of filtering the central frequencies for 49 times.
3. In the loop: first reshape the data on the columns to the n,n,n dimension.
4. In the loop: perform *FFT* to the reshaped 3D data using `fftn`, since the data are reshape into 3D dimension.
5. In the loop: apply filter to the shifted signal in frequency domain.
6. In the loop: apply inverse shift and transfer to retrieve the original signal `Unf`.
7. In the loop: compute isosurface data from absolute value of `Unf` using `[f,v] = isosurface`. Then retrieve the coordinate of each dimension.
8. Out side the loop: plot the 49 steps path of the submarine using data from each coordinate.

Algorithm 2: Filtering Around the Central Frequencies

Define the filter in three dimension
Define variable loci to be a zero array with dimension of 49*3
for $j = 1 : 49$ **do**
  Reshape data on the jth columns to n*n*n dimension
  Perform FFT by using `fftn` function to the reshaped 3D data `Un`
  Use `fftshift()` to shift the data into correct order then apply the filter to the signal in frequency space
  Apply inverse shift `ifftshift` to restore the original order of data on the frequency domain.
  Apply inverse Fourier transfer `ifftn` to retrieve the original signal `Unf`
  Retrieve the coordinates with isosurface data using `[f,v] = isosurface`
end for
Plot each location according to their coordinates in a 3D space using `plot3`.

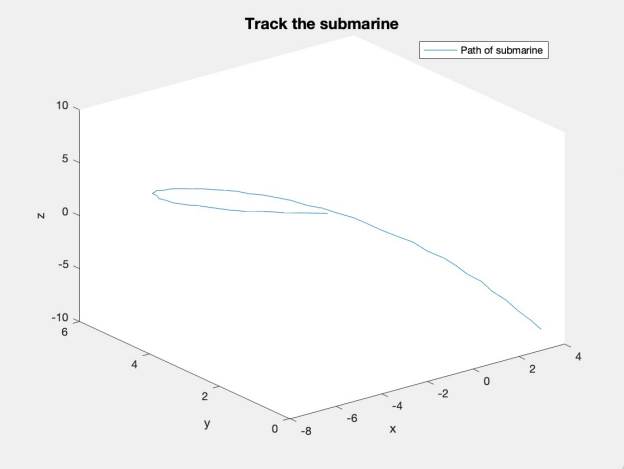

Figure 1: Path of the Submarine

3.4 Aircraft tracking table

In order to track the submarine using the P-8 Poseidon sub-tracking Aircraft, the first step is to locate the final position of the submarine. This can be done by tracing the path of the submarine and locates the final position using the last row of data.

1. Create table to record the x and y coordinates of the path of submarine without z coordinate because the aircraft only need to locate its position in the desire altitude.

4 Computational Results

4.1 Central Frequency

Perform Algorithm 1. Based on the results, the coordinates of the central frequency generated by the submarine is given by {`5.0477,-6.9961,2.0461`}.

4.2 Path of the Submarine

After performing the Algorithm 2, the outcome is a 3D graph of the path of submarine using `plot3`, which is shown in `Figure 1`.

X coord	Y coord
-5.0881	0.8169

Table 1: Location of the Submarine

4.3 Location of the P-8 Poseidon sub-tracking Aircraft

Executes the table of the x and y coordinate correspond to the last position of submarine, which gives the result shows in `Table 1`.

5 Summary and Conclusions

This research paper studies on the topic of tracking the submarine using the noisy acoustic data collected over 24 hours emits by the submarine in the Puget Sound. By using computational tools including implement averaging over the spectrum and filtering around the center frequency. In order to employ these tools, this paper prepares the theoretical background which includes the Fourier transform and filter with Gaussian function. After implement the algorithm, the paper documents the computational results which includes the value of central frequency, the graph indicates the path of the submarine, and the appropriate location of the P-8 Poseidon sub-tracking Aircraft in order to track the submarine. The significance of this research is the implementation indicates that one can uses averaging and filtering to process trunk of noisy data in the real world, and it is possible to recover data with distribution. This is a topic worthy of study and is a great tool for one to understand the background theology and algorithms to perform a cleanup on a noisy data set.

References

[1] Jose Nathan Kutz. *Data-driven modeling & scientific computation: methods for complex systems & big data*. Oxford University Press, 2013.

Appendix A MATLAB Functions

List of the important MATLAB functions with brief implementation explanation that are used to execute the algorithm.

- `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.
- `zeros(n,n,n)` returns a n by n by n array of zeros.
- `reshape(A,n,n,n)` reshape A into a n by n by n array.
- `M = max(A,[],'all')` returns the maximum element of array A.
- `fftn(X)` uses the Fast Fourier Transform algorithm to return the multidimensional Fourier Transform of an N-D array.
- `fftshift(A)` shift the zero frequency component to center of the spectrum
- `[f,v] = isofurface()` extract isosurface data from the volume data and returns the faces and vertices of the array.
- `ifftn(A)` returns the multidimensional inverse discrete Fourier Transform, using FFT, of A with same size.

- `ifftshift(A)` shift the zero-frequency-shifted Fourier Transform X back to the original transform output.
- `plot3(X,Y,Z)` plots the corresponding coordinates in 3D space.
- `T = table(var1,...,varN)` creates a table using the input variables var1,...,varN.

Appendix B MATLAB Code

The MATLAB codes uses to execute the algorithm 1 and 2.

```
% Clean workspace
clear all; close all; clc

load subdata.mat % imports data as the 262144*49 (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);

% compute transfered noisy signal using a loop
sum_ave = zeros(n,n,n);

for j=1:49
    Un(:,:,:)=reshape(subdata(:,j),n,n,n);
    M = max(abs(Un),[],'all');
    Utn = fftn(Un);
    sum_ave = sum_ave + Utn;
    close all, isosurface(X,Y,Z,abs(Un)/M,0.7)
    axis([-20 20 -20 20 -20 20]), grid on, drawnow
    pause(0.01)
end

% averaging the spectrum and determine the frequency signature
ave = abs(fftshift(sum_ave))/49;
M = max(abs(ave),[],'all');
close all,
[f,v] = isosurface(Kx,Ky,Kz,abs(ave)/M,0.7);
axis([-20 20 -20 20 -20 20]), grid on, drawnow

cfreq = mean(v);
kx = cfreq(1);
ky = cfreq(2);
kz = cfreq(3);
```

Listing 1: Code to perform algorithms 1

```
%%
% filter the data around the central frequency
tau = 0.2;

filter_x = exp(-tau*(Kx-kx).^2);
filter_y = exp(-tau*(Ky-ky).^2);
filter_z = exp(-tau*(Kz-kz).^2);
filter_center = filter_x.*filter_y.*filter_z;

loci = zeros(49,3);
for j=1:49
    Un(:,:,:)=reshape(subdata(:,j), n,n,n);
    Utn = fftn(Un);
    Unft = filter_center.*fftshift(Utn);
    Unft = ifftshift(Unft);
    Unf = ifftn(Unft);
    M = max(abs(Unf),[],'all');
    [f,v] = isosurface(X,Y,Z,abs(Unf)/M,0.7);
    loci(j,:) = mean(v);
end

% plot the path of submarine
plot3(loci(:,1),loci(:,2),loci(:,3));
legend ('Path of submarine','Location','best');
xlabel('x');
ylabel('y');
zlabel('z');
title("Track the submarine", 'Fontsize', 14)

% table of x and y lociinates of P-8 Poseidon subtracking aircraft position
T = table(loci(49,1),loci(49,2));
```

Listing 2: Code to perform algorithms 2