AMATH 482 Homework 1

Tommy Dong

January 24, 2020

Abstract

This project is about dealing with signals in real life with Fourier Transform and other techniques. In particular, one was offered with a data that contains both the actual signal and white noises that may disturb our search for the real frequency signature. This paper illustrates the the background, goal, theoretical background, implementation result and conclusion.

1 Introduction and Overview

One is given a real life data about the acoustic information from the Puget Sound, and the goal is to identify the frequency signature of the submarine and hunt it down afterwards. During the process, there are several things we need to accomplish in order to achieve our goals.

1.1 Data Description

The data in the file is a 262144×49 matrix, which then we transform into a $64 \times 64 \times 64 \times 49$ matrix. For the transformed matrix, the first three 64 elements represent the x, y, z coordinates in the certain area, while 49 represent 49 different time when the data is collected. The values in the matrix are the acoustic data for certain position in the area for specific time.

1.2 Goal

In this project, we have three main goals:

1. Identify the frequency signature of the submarine.
2. Denoise the original data in order to draw the movement path of the submarine.
3. Return the coordinates for the submarine in 49 realizations.

2 Theoretical Background

2.1 Fourier Transformation

This is the Fourier Transform equation. It can change a function of time ,x, into a function of frequency, k.

$$\hat{f}(k) = \frac{1}{\sqrt{2\pi}} \int_{-\infty}^{\infty} f(x) e^{-\mathrm{i}kx}\, dx \tag{1}$$

Below is the inverse Fourier Transform. It did the opposite as Fourier Transform, converting a function of frequency back to a function of time or space.

$$f(x) = \frac{1}{\sqrt{2\pi}} \int_{-\infty}^{\infty} \hat{f}(k) e^{-\mathrm{i}kx}\, dk \tag{2}$$

The Fourier Transform above is only suitable for infinite domains. When only having a limited range of x values, Fourier introduced a concept of representing $f(x)$ by a trigonometric series of sines and cosines:

$$f(x) = \frac{a_0}{2} + \sum_{i=1}^{\infty} (a_n \cos nx + b_n \sin nx) \quad x \in (-\pi, \pi]. \tag{3}$$

The Discrete Fourier Transform is just like the Fourier Transform but truncated at some maximum frequency to reflect this potential shortcoming.

$$\hat{x}_k = \frac{1}{N} \sum_{n=0}^{N-1} x_n e^{\frac{2\pi i k n}{N}} \tag{4}$$

2.2 Gaussian Equation

Gaussian equation is the filter in this project for its bell-shape property can assist to amplify the real signal and remove the influence of noises if filter around the center signal.

$$f(x) = e^{-\tau(x-x_0)^2} \tag{5}$$

3 Algorithm Implementation and Development

1. Averaging frequencies
   The real signal will also be the same shape in spatial domain, but may shift its position along with time changes. White noises, however, its frequency varies with time and tend to have a zero mean. So averaging frequencies is to take several realizations' frequencies, do the Fourier Transform and then average the different realizations in the frequency domain. As for the issue of different positions for the signal, this will not be a problem for the signal shifted in the time domain will look the same in the frequency domain.

   For this project, we use the averaging technique and extract the frequency coordinates of the maximum, which is frequency signature of the submarine in frequency domain.

2. Filtering frequency
   Signals can be difficult to detect with the disturbing of noises. So when we know the center frequency, we introduce a spectral filter around the center frequency to remove the influence of noises in order to get a better signal.

   In our case, as we have already found out the center frequency, we utilize a 3D Gaussian function around the center frequency as the filter to clarify the signal of the submarine. We first transform each of the 49 realizations into frequency. Then we multiply the filter in and transform them back into the spatial domain. And we can look at the maximum point for each realization, connecting them to form the path of the submarine across time.

4 Computational Results

4.1 Frequency Signature of the submarine

The coordinates of the frequency signature of submarine is $[5.63, -6.56, 2.50]$.

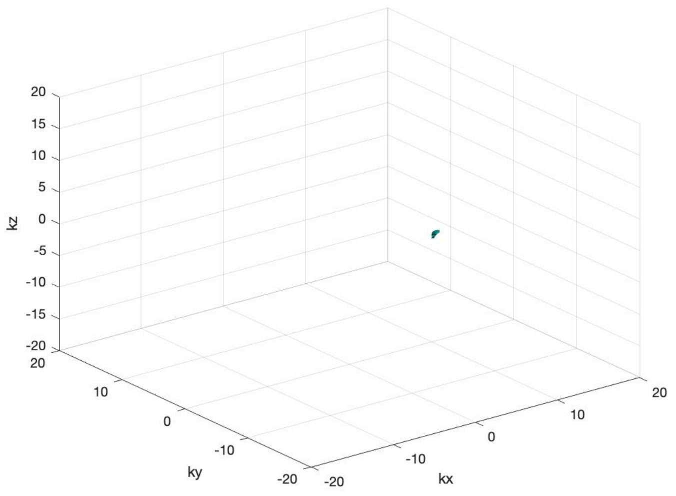

Figure 1: The frequency signature of the submarine in frequency domain

4.2 The path of the submarine

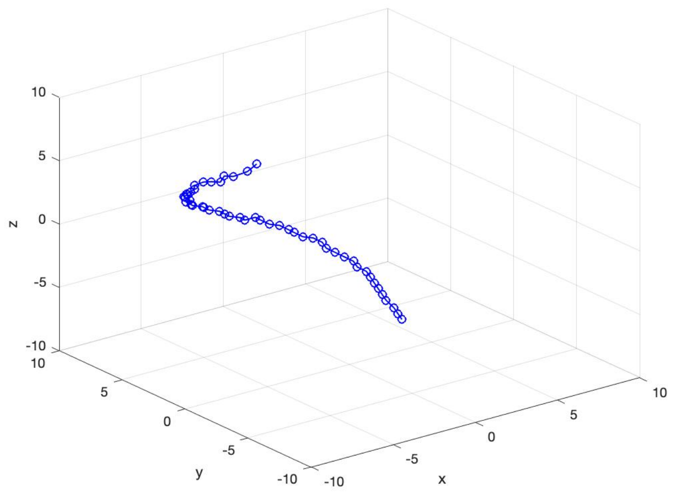

Figure 2: The path of the submarine across time

4.3 The x,y coordinates of the submarine

x coor	y coor
-4.6875	1.2500

Table 1: Final x,y coordinates of the submarine

5 Summary and Conclusions

As we managed to averaging the frequencies, we got the center frequency in the frequency domain, as Figure 1 showed. With the information of the center frequency, we applied a filter function around the center frequency and we successfully removed the effect of white noises. With the clean signal of submarine in each realization, we tracked down the submarine and found its trajectory was spiraling, like Figure 2. With its path, we knew its location across the time and we obtained the final x,y position of the submarine for our P-8 to subtracking, the coordinates is in Table 1.

From this project, Fourier Transform is essential when identifying the unique signal of the submarine, and use this frequency signature to locate the submarine in the area. With certain techniques like averaging and filtering we implemented in this project, the problem of white noises and the shift of the signal itself are perfectly solved.

Appendix A MATLAB Functions

Add your important MATLAB functions here with a brief implementation explanation. This is how to make an **unordered** list:

- `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.
- `B = reshape(A,sz)` reshapes `A` using the size vector, `sz`, to define size(B). For example, `reshape(A,[2,3])` reshapes `A` into a 2-by-3 matrix. `sz` must contain at least 2 elements, and `prod(sz)` must be the same as `numel(A)`.
- `[M,I] = max(A,[],'all','linear')` returns the linear index into `A` that corresponds to the overall maximum value in `A`.
- `[I1,I2,...,In] = ind2sub(sz,ind)` returns `n` arrays `I1,I2,...,In` containing the equivalent multidimensional subscripts corresponding to the linear indices ind for a multidimensional array of size `sz`. Here `sz` is a vector with n elements that specifies the size of each array dimension.
- `Y = fftn(X)` returns the multidimensional Fourier transform of an N-D array using a fast Fourier transform algorithm. The N-D transform is equivalent to computing the 1-D transform along each dimension of `X`. The output `Y` is the same size as `X`.
- `Y = fftshift(X)` rearranges a Fourier transform `X` by shifting the zero-frequency component to the center of the array.
- `X = ifftshift(Y)` rearranges a zero-frequency-shifted Fourier transform `Y` back to the original transform output. In other words, `ifftshift` undoes the result of `fftshift`.
- `X = ifftn(Y)` returns the multidimensional discrete inverse Fourier transform of an N-D array using a fast Fourier transform algorithm. The N-D inverse transform is equivalent to computing the 1-D inverse transform along each dimension of `Y`. The output `X` is the same size as `Y`.

Appendix B MATLAB Code

Add your MATLAB code here. This section will not be included in your page limit of six pages.

```
%% Project 1
% Clean workspace
clear all; close all; clc
format short
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);

%% Avergaing Frequencies
Unave = zeros(64,64,64);
for j=1:49
    Un(:,:,:)=reshape(subdata(:,j),n,n,n);
    Unfftn=fftn(Un);
    Unave = Unave + Unfftn;
end
Unave = abs(fftshift(Unave))/49;
[M,idx]=max(abs(Unave),[],'all','linear');
[y0,x0,z0]=ind2sub(size(Unave),idx);
v=[x0,y0,z0];
v=v.*(20/64)-10
figure(1)
isosurface(Kx,Ky,Kz,abs(Unave)/M,0.7)
axis([-20 20 -20 20 -20 20]), grid on, drawnow,
xlabel('kx'),ylabel('ky'),zlabel('kz')

%% Filter around center frequency to find the path of submarine
tau=0.2;
coor=zeros(49,3);
filter = exp(-tau*((Kx-v(1)).^2 +(Ky-v(2)).^2+(Kz-v(3)).^2));
% filter = exp(-tau*((Kx-5).^2 +(Ky+7).^2+(Kz-2).^2));
for k=1:49
    Un(:,:,:)=reshape(subdata(:,k),n,n,n);
    Unfftn=fftn(Un);
    Unfftn=filter.*fftshift(Unfftn);
    Unfftn=ifftshift(Unfftn);
    Unf=ifftn(Unfftn);
    [M,idxn] = max(abs(Unf),[],'all','linear');
    [b,a,c]=ind2sub(size(Unf),idxn);
    coor(k,1)=a*20/64-10;
    coor(k,2)=b*20/64-10;
    coor(k,3)=c*20/64-10;
end
figure(2)
plot3(coor(:,1),coor(:,2),coor(:,3),'bo-','Linewidth',1)
axis([-10 10 -10 10 -10 10]), grid on, drawnow
xlabel('x'),ylabel('y'),zlabel('z')
%% x,y coordinates for the submarine
coor(:,[1,2])
```