AMATH 582 Homework 1

Junzhao Liu

January 24, 2020

Abstract

The goal of this homework is hunting a moving submarine in the Puget Sound. By using noisy acoustic data, we are trying to use what we learned from the previous three lectures to find the path of the submarine.

1 Introduction and Overview

There are three questions in this assignment, first of all, we need to determine the frequency signature generated by submarine. Second, denoise the data and find the path of the submarine. Lastly, find the location to send P-8 Poseidon subtracting aircraft

1.1Maximum Frequency Point

Since the noisy signals are typically white noise, which means they affect the frequencies the same. In order to minimize the effect, we can average the spectrum to determine the frequency signature generated by submarine. Based on the code given, set up a for loop runs for 49 times. Since we need apply a higher Dimensional Fourier transforms in the homework to analyze 3D data, we will use build -in function FFTN instead. After applying FFTN we need to apply FFTSHIFT to get the data in the right order and then add them together. Therefore, we can find the maximum frequency point.

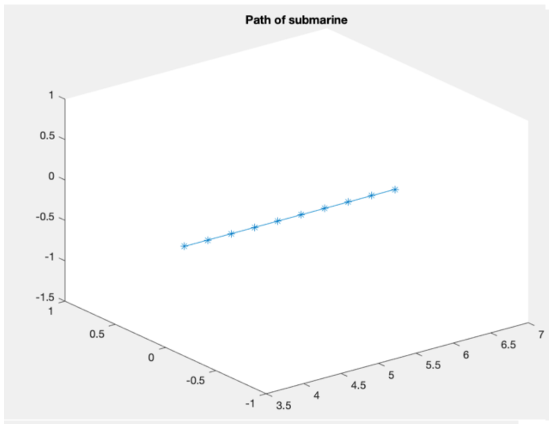
1.2Filtering the frequency

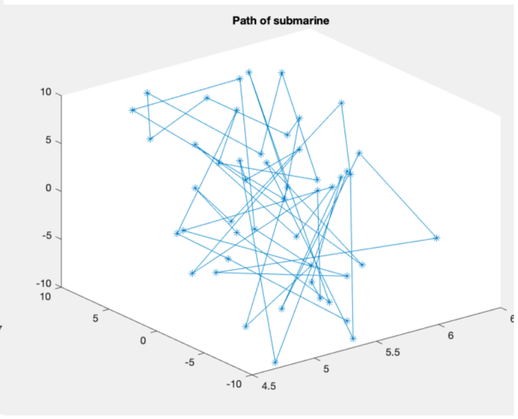
Follow what we did in class, firstly I plot the filtered signal in frequency domain. we should first find UT which is the FFT of given data and then find the shift Fourier transform of UT, named as UTSHIFT. Use filter times UTSHIF to obtain the filtered signal. I tried different filters including

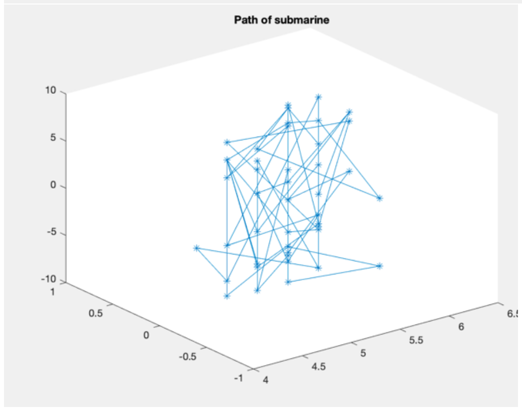
$$filter = exp(-tau * (Kx - x_in).^2 + (Ky - y_in).^2 + (Kz - z_in).^2);$$

$$filter = exp(-tau * (Kx - X_value).^2);$$

$$filter = exp(-tau * (Kx - X_value).^2 + (Ky - Y_value).^2);$$







the first one gives us the most efficient result. Therefore, I decided to use the first filter.

Then I tried different value of tau, no big differences have been detected.

2 Theoretical Background

As we learned from lectures, Fourier transform takes a function of space or time and convert it to a function of frequencies k, basically we need transfer the noisy acoustic data in the package to frequencies. By doing so, we need apply the Discrete Fourier transform (DFT), in order to achieve the goal faster, we need apply MATLAB build-in functions such as FFT Transform. After that, we need to identify a filter which can help us locate the submarine. In the end, we can graph the pathway of the submarine.

3 Algorithm Implementation and Development

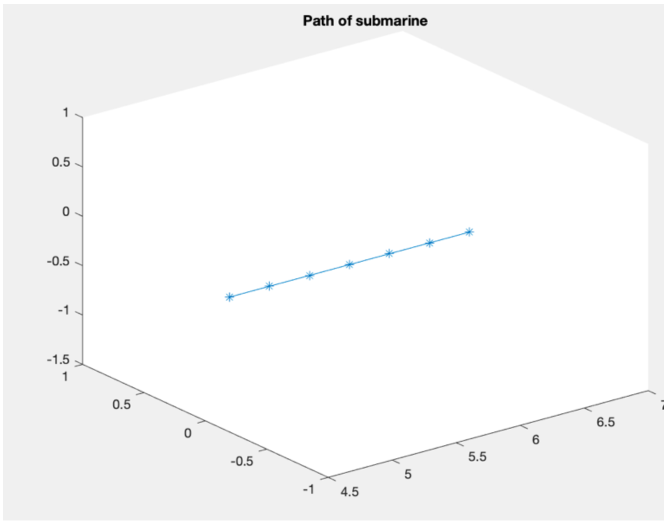
Add your algorithm implementation and development here. See Algorithm 1 for how to include an algorithm in your document. This is how to make an ordered list:

1. Find the center frequency
2. Use center frequency to denoise the data and find the path of submarine
3.find the last location of submarine.

4 Computational Results

The frequency signature is [x,y,z]=[-4.7124, 3.1416, -7.854]

Path of submarine



Final location of submarine (X= 5.3125 y = 0 z = -0.3125)

Summary and Conclusions

Follow the steps from our lectures, we successfully found the path of submarine and its final location.

Appendix A MATLAB Functions

[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

```
% 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);
ave = zeros(n,n,n);
 for i = 1:49
     ave = ave+fftshift(fftn(reshape(subdata(:,i),n,n,n)));
 end
 ave = abs(fftshift(ave))/49;
 maxave = max(abs(ave),[],'all');
 [x,y,z]=ind2sub(size(ave),find(abs(ave)==maxave));
 x_in = Kx(x,y,z);
 y_in = Ky(x,y,z);
 z_in = Kz(x,y,z);
 X_value = X(x,y,z);
 Y_value = Y(x,y,z);
 Z_value = Z(x,y,z);



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



   %Filter data
 tau = 0.5;
 k0 = 0;
 filter = exp(-tau*(Kx-x_in).^2+(Ky-y_in).^2+(Kz- z_in).^2);
 %filter = exp(-tau*(Kx-X_value).^2);
 %filter = exp(-tau*(Kx-X_value).^2+(Ky-Y_value).^2);
 u= 0;
 directions= zeros(3,49);
 for i = 1:49
     u =reshape(subdata(:,i),n,n,n);
     ut = fftn(u); %fft of u
     utshift = fftshift(ut);% shift Fourier transform ut
     unft = filter.* utshift;
     unf = ifftn(unft);
     signal = ifftshift(unft);
     maxu = max(abs(signal),[],'all');
     [x,y,z]=ind2sub(size(signal),find(abs(signal)==maxu));
     x1 = X(x,y,z);
```

```
    y1 = Y(x,y,z);
    z1 = Z(x,y,z);
    directions(1,i)= x1;
    directions(2,i)= y1;
    directions(3,i)= z1;
end
plot3(directions(1,:),directions(2,:),directions(3,:),'-*')
title('Path of submarine')

x_last = directions(1,49);
y_last = directions(2,49);
z_last = directions(3,49);
final = [x_last,y_last, z_last]
```

.