Amath482 Homework1

Kexin Jiao

Abstract:

This assignment mainly focuses on the practice of the Fast Fourier Transform(and its inverse), Filtering, and Averaging in 3D dimension to solve the actual problem: hunting for a submarine by analyzing a data of spectrum recording of noisy acoustics.

Introduction and overview:

The homework is about hunting for a submarine in the Puget Sound using noisy acoustic data. It is a new submarine technology that emits an unknown acoustic frequency that we need to detect. Using a broad spectrum recording of acoustics, data is obtained over a 24-hour period in half-hour increments. Unfortunately, the submarine is moving, so its location and path need to be determined. Our goal is to locate the submarine, find its trajectory using the acoustic signature, and identify the acoustic admissions of this new class of submarine. A 262144*49 dataset is given, which includes 49 columns for acoustic measurements over a 24-hour span at half-hour increments in time. We are asked to perform the following:

1. Through averaging of the spectrum, determine the frequency signature (center frequency) generated by the submarine.
2. Filter the data around the center frequency determined above in order to denoise the data and determine the path of the submarine. Use plot3 to plot the path of the submarine once you have it.
3. Where should you send your P-8 Poseidon subtracking aircraft? Give the x and y coordinates in a table to follow the submarine.

Theoretical background:

Fourier Transform

Suppose we are given a function $f(x)$ with $x \in R$. The Fourier Transform of f(x),written $f\_hat(k)$, is defined by the formula

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

Inverse Fourier Transform

If we are given $f\_hat(k)$ and want to recover $f(x)$, we can use the Inverse Fourier Transform

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

Discrete Fourier Transform(DFT)

If we are given a a function at a discrete set of points, we can never know about extremely high frequency between these points.The DFT is similar to the Fourier series but truncated at some maximum frequency to reflect this potential shortcoming. Suppose we are given a sequence (or vector) of N values that are a function sampled at equally-spaced points {x0, x1, x2, . . . , x(N-1)}. The Discrete Fourier Transform is a sequence of numbers:

$$x\_hat\_k = \frac{1}{N}\sum_{n=0}^{N-1} x\_n * e^{\frac{2\pi i k n}{N}}$$

Fast Fourier Transform(FFT)

The FFT does faster than DFT: the total complexity of DFT is O(N^2), while the total complexity of FFT is only O(Nlog(N)). The idea behind FFT is that if we have a sequence of length N, we can split the DFT into two of length N/2 and repeat the process over and over again, referred to as a divide and conquer algorithm. MATLAB has a built-in fft function to perform the process.

Filtering and Averaging

Signals are never clean in the real world. There might even be other sources of electromagnetic energy at the desired frequency, which results in a noisy signal. Usually the noise is white noise that affects all frequencies the same. In order to add white noise to the idealized MATLAB signal, we can add a (complex) random number to each Fourier coefficient, drawing the random values from the normal distribution with mean 0 and standard deviation 1:

$$noise = 1;$$

$$utn = ut + noise * (normrnd(0,1,1,n), + li * normrnd(0,1,1,n))$$

Given that white noise was modeled by adding a normally distributed random variable with zero mean and unit variance to each Fourier component, if we average over many realizations (in frequency space), the noise from each realization will cancel out and we will get something close to the true signal.

(Reference: Amath482 lecture notes from Professor Jason Bramburger)

Algorithm implementation and development:

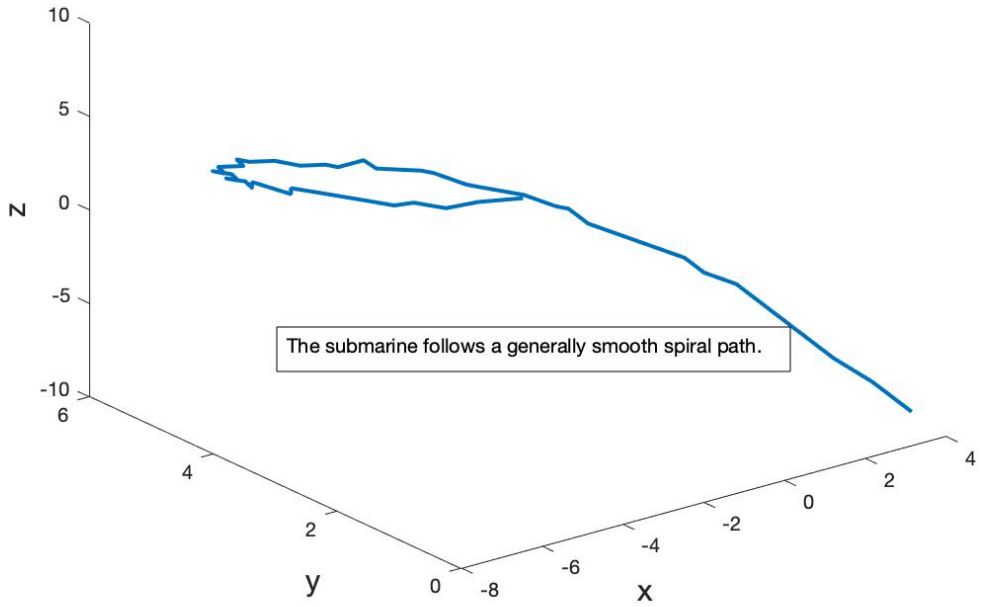
For the starter code, I made the frequency component K = 2*pi/(2*L))*[0:(n/2 - 1) -n/2:-1]. Specifically, I rescaled the frequencies by 2pi/2L since the FFT assumes 2pi periodic signals. I reorganized it by fftshift function. Also, I used the meshgrid function to return 3D coordinates because the data of acoustic frequency given in the context is in 3D. To find the center frequency generated by the submarine, I averaged the spectrum. Making a for-loop with

49 sets of data(49 iterations), I first reshaped the given dataset(subdata.mat) into the form of a 64^3 matrix. I then employed Fast Fourier Transform(fftn function for 3D) to transform the data into the frequency domain and summed the frequencies up. After rearranging the outputs of fftn by the fftshift function, I averaged the frequency. To be specific, I used the max function to find the maximum frequency and used the index of the maximum to locate the xyz position of the center frequency. To filter the noise around the center frequency, I made the loop again. Within the loop, I reshaped the dataset and applied Fast Fourier Transform to get the data of frequency. After I filtered the noise with the Gaussian filter function in the frequency domain, I rearranged the output by the ifftshift function and converted the data of frequency to the data of time by the Inverse Fast Fourier Transform with the use of ifftn function. To find the path of the submarine, I employed the max function to get the maximum frequency and used the find function to get the linear indices of the data. Then, I used the ind2sub function to return arrays of xyz vectors containing the equivalent multidimensional subscripts corresponding to the linear indices for a multidimensional array. I respectively extracted x,y,z vectors(the positions of x,y,z) from 3D arrays by the squeeze function. Finally, I plotted the path into 3D space using plot3 function.

Computational results:

1. The central frequency is at (5.3407,-6.9115,2.1991).
2. We can see from the figure that the path of the submarine is a smooth spiral with some small jumps.

Path of the submarine



3.

	x	y
1	3.1416	0
2	3.1416	0.3142
3	3.1416	0.6283
4	3.1416	1.2566
5	3.1416	1.5708
6	3.1416	1.8850
7	3.1416	2.1991
8	3.1416	2.5133
9	3.1416	2.8274
10	2.8274	3.1416

	x	y
11	2.8274	3.4558
12	2.5133	3.7699
13	2.1991	4.0841
14	1.8850	4.3982
15	1.8850	4.7124
16	1.5708	5.0265
17	1.2566	5.0265
18	0.6283	5.3407
19	0.3142	5.3407
20	0	5.6549

	x	y
21	-0.6283	5.6549
22	-0.9425	5.9690
23	-1.2566	5.9690
24	-1.8850	5.9690
25	-2.1991	5.9690
26	-2.8274	5.9690
27	-3.1416	5.9690
28	-3.4558	5.9690
29	-4.0841	5.9690
30	-4.3982	5.9690

	x	y
31	-4.7124	5.6549
32	-5.3407	5.6549
33	-5.6549	5.3407
34	-5.9690	5.3407
35	-5.9690	5.0265
36	-6.2832	5.0265
37	-6.5973	4.7124
38	-6.5973	4.3982
39	-6.9115	4.0841
40	-6.9115	3.7699

	x	y
41	-6.9115	3.4558
42	-6.9115	3.4558
43	-6.9115	2.8274
44	-6.5973	2.5133
45	-6.2832	2.1991
46	-6.2832	1.8850
47	-5.9690	1.5708
48	-5.3407	1.2566
49	-5.0265	0.9425

Summary and conclusions:

For this assignment, I applied FFT(fftn and fftshift functions) to average the center frequency. I used Gaussian filtering function to denoise the data around the center frequency in frequency space and converted back to the time domain by inverse FFT(ifftn and ifftshift functions). To determine the path of submarine and send subtracking aircraft, I employed the max, find, ind2sub, and squeeze functions to track the positions of submarine in 3D space.

MATLAB function used and brief implementation explanation:

-load: load data(variables) from file into workspace
-linspace: generate linearly spaced vector
-meshgrid: returns 3D(in this case) grid coordinates defined by the vectors xyz
-zeros: create array of all zeros
-reshape: reshape(reorganize) array by a size vector
-fftn: return the multidimensional Fourier Transform of an N-D array using a Fast Fourier Transform algorithm
-fftshift: rearrange a Fourier Transform by shifting the zero-frequency component to the center of the array
-max: ([A,I] = max())return the maximum value and its index
-ifftn: return the multidimensional Discrete Inverse Fourier transform of an N-D array using Fast Fourier Transform algorithm
-ifftshift: rearrange a zero-frequency-shifted Fourier Transform back to the original transform output, undo the result of fftshift
-find: return the index of element(s)
-ind2sub: convert linear indices to subscripts
-squeeze: extract vectors(arrays) from (3D)arrays
-plot3: plot coordinates in 3D space
-table: create a table from inputs
(Reference: MATLAB search helps)

MATLAB codes:

```
% 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);
```

```
% problem1
ave = zeros(n,n,n);
for j=1:49
    Un(:,:,:)=reshape(subdata(:,j),n,n,n);
    Utn=fftn(Un);
    ave = ave+Utn;
end
ave = abs(fftshift(ave))/49;
[A,I] = max(ave(:));
center = [Kx(I),Ky(I),Kz(I)];
%isosurface(X,Y,Z,abs(ave)/max(abs(ave), [], 'all'),0.7)

% problem2
tau = 0.5;
filter = exp(-tau*((Kx-center(1)).^2+(Ky-center(2)).^2+(Kz-center(3)).^2));
indice = zeros(1,49);
for k = 1:49
    Un(:,:,:)=reshape(subdata(:,k),n,n,n);
    Utn = fftn(Un);
    Unft = filter.*fftshift(Utn);
    Unf = ifftn(ifftshift(Unft));
    maximum = max(abs(Unf(:)));
    indice(k) = find(maximum == abs(Unf));
end
sz = [64,64,64];
[Xvec,Yvec,Zvec] = ind2sub(sz,indice);
xp = Kx(Xvec,Yvec,Zvec);
yp = Ky(Xvec,Yvec,Zvec);
zp = Kz(Xvec,Yvec,Zvec);
xxp = squeeze(xp(49,:,49));
yyp = squeeze(yp(:,49,49));
zzp = squeeze(zp(49,49,:));
plot3(xxp,yyp,zzp,"LineWidth",2)
xlabel("x","fontsize",15)
ylabel("y","fontsize",15)
zlabel("z","fontsize",15)
title("Path of the submarine","fontsize",20)
annotation('textbox','String','The submarine follows a generally smooth spiral path.')

% problem3
T = table(xxp.',yyp);
```