AMATH582 HW1 AN ULTRASOUND-LIKE PROBLEM

SHUANG WU

Jan. 20th

Abstract

For the first homework, I would like to deal with an ultrasound-like problem. I have a lot of data that comes from the ultrasound test for my dog Fluffy. My dog swallowed a marble by accident so I need to get the marble out as soon as possible. The data was noisy because my dog kept moving while the test running. I will try to apply the Fourier transformation and de-noisy procedure to keep my dog alive!

1 INTRODUCTION

Yesterday, when I was not at home, I left my dog Fluffy alone. I found that he swallowed a marble when I came back. I took him to the pet hospital to take the marble out. The doctor wanted to use the ultrasound to test where the ball was and break the ball after knowing the exact location. Unfortunately, Fluffy kept moving during the test and the data came out with lots of nosies (Show in the Figure 1 later). The doctor had no idea about how to deal with the marble and asked me to try to find the exact location, otherwise my dog would die.

2 THEORETICAL BACKGROUND

Based on what I have known, I need to use the Fast Fourier transformation and build a filter to find the real signal that is useful for me and de-noise unuseful signals.

2.1 *Fast Fourier Transformation (FFT)*

Fast Fourier transformation will be used. Because the dog keeps moving, the signal has noises. We want to transfer the data to frequency domain or spectral domain. By examining the frequency, we will find where the signal is. Because the sample mean of the noises, which is calculated by averaging the sum should always be approximate to zero, the signal will stay in the same condition even if the dog keeps moving in frequency domain.

So I use the Fourier Transform equation for the noisy data set by the equation:

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

After I transformed each row of the data, I would like to add all data up and then take the average of the sum. After these steps, I can figure out the frequency and the exact location for the marble. Then, I will build filter that can filter out the related frequency and then transform back to the spatial domain, which will help me know how the marble has moved. To transfer the data back to the spatial domain, I need this equation:

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

In applying the FFT in Matlab, I need to do a 3-dimensional transformation instead of a 1-D transformation. That is the reason why before dealing with the data. I reshaped the normal data to a 3-D data set with size $64 * 64 * 64$.

To make sure that I can solve the problem in a fast speed, I created $2^6 = 64$ points. Also, because the Fourier transformation is defined on 2π domain, I created a k vector that defined in 2π domain to associate with spatial domain.

2.2 *Filter*

After figuring out the location of the true frequency of the marble, I need to use a filter that can filter the noisy data out so only the true frequency will be left. Finishing all previous steps, I can transform the data back to the original form and know how the marble has moved.

The normal filter I use is the Gaussian filter with the normal equation:

$$Filter = e^{-0.2*k^2}$$

.
And I expand the equation to 3 dimensions, which will be this formula:

$$Filter = e^{-0.2*((kx-kx_0)^2+(ky-ky_0)^2+(kz-kz_0)^2)}$$

3 ALGORITHM IMPLEMENTATION

For the algorithm in Matlab, I have two main parts that are worthy to be considered. The first part is finding the frequency location, and the second one is finding the marble location.

3.1 *Finding the frequency of marble location*

1. Built up the basic term.
   I first need to set up the spatial domain and Fourier models, which are 15 and 64. I also need x, y, z vectors in the spatial domain. Because the FFT is on the 2π domain, I need define k vector that transfers the spatial domain to 2π domain. After getting all of these vectors ready, I build my 3-D models, the X, Y, Z 3-D model for real data and the Kx, Ky, Kz 3-D model for FFT data by using the function `meshgrid` in Matlab.

2. Calculate the average.
   Because I have 20 different data and the data sets are based on different time, I need to average each row of the data. Thus, I use a `for` loop that can go through each row of the data. In the `for` loop, I first reshape each row of the data into 3-D, whose dimension is $64 * 64 * 64$. After reshaping the data, I apply the FFT with 3-D using the function `fftn` in Matlab. After this, I sum up the whole data to a vector named `total`, which contains the sum of all the data. At the end of the for loop, I have the sum of all the FFT data in the frequency domain. Then I average the sum. But while I taking the average, I also take the fftshift and absolute value for the sum. Because the data will include imaginary part after applying FFT, and if I want to plot the data, I need to do the fftshift to make the plots in the right position.

3. Normalize the frequency and figure out the exact location.
   After I have my frequency data, I normalize the data by using the Matlab function `ind2sub` to find the location, Kx, Ky, Kz, of the maximum values in frequency domain.

3.2 *Finding the marble location*

1. Built the filter.
   Having the location of the frequency, I build my filter using the equations mentioned in section 2.2. This filter will filter out the actual movement of the marble.

2. Apply the filter to data
   I define another `for` loop that takes each row of the data and then transforms the data to 3-D ($64 * 64 * 64$). Then, I make the FFT transfer in 3-D and also do the fftshift to make sure the data in the same domain with my filter. Having the data ready, I take multiplication of the filter to the data one by one (using $.*$). Then I transform the data back to the original spatial domain by using `ifftn` in 3-D.

3. Find the marble location at each time and plot.
   In the `for` loop, after I did the inverse transform data, I take the absolute value at first to make sure there is no imaginary part. Then, I use the same way as I used before to find locations of the marble, and store all the locations' value of X, Y, Z for each row in a $20 * 3$ matrix. Also, I use the function `isosurface` to plot the trajectory of the marble. Outside the `for` loop I use `plot3` to plot a clearer plot that describes how the marble moves.

4 COMPUTATION RESULTS

At the beginning, I had a plot like this:

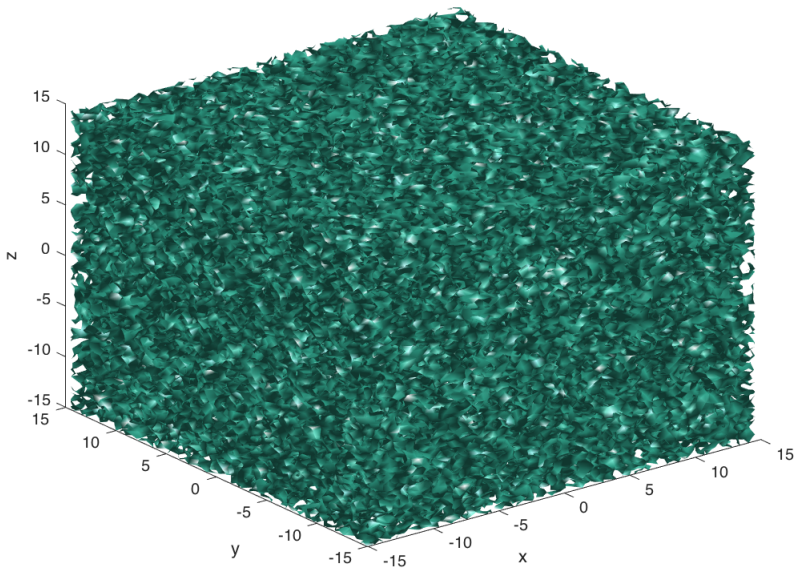

Figure 1: Original data at the last time(last row)

The marble was at somewhere of the huge cube, and now let's try to average the frequency.

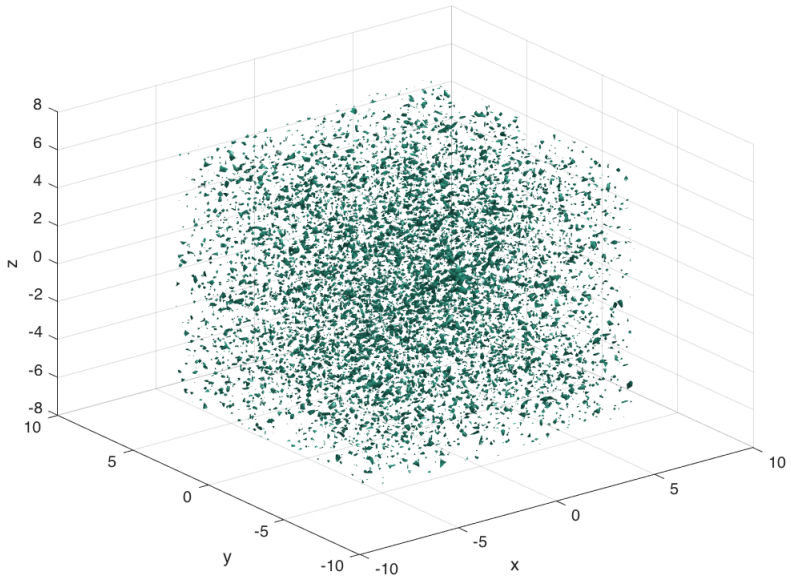

Figure 2: Averaging frequency with isovalue=0.4

Figure 2 is the plot in frequency domain after I did the averaging, it looks better than Figure 1, but still, it is hard to find the exact location of that marble. Now, I need to increase the isovalue, which will affect the solution. I try values from 0.4 to 0.7 then I got a plot that almost has only one frequency

of the marble.

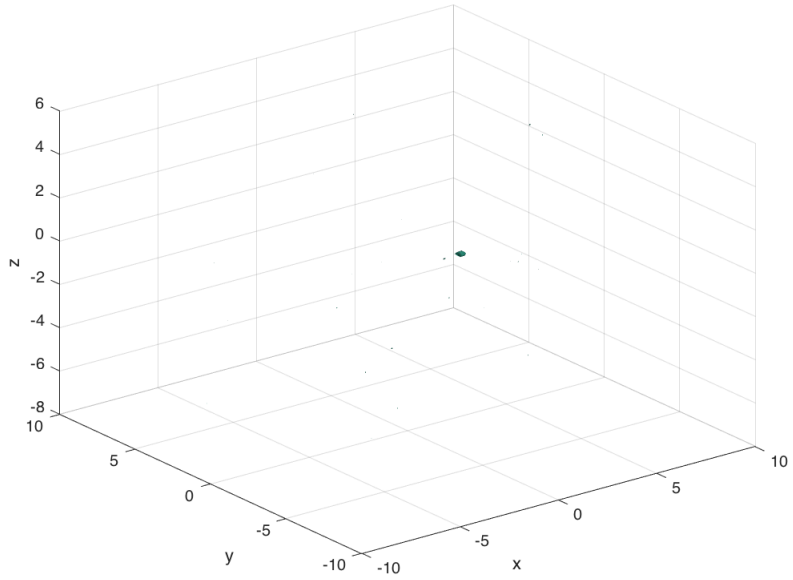

Figure 3: Averaging frequency with isovalue=0.7

After I found the marble, I know the center of the right frequency. Then I build the filter using the strategy the I mentioned in Section 3 to find the exact trajectory of this marble, which looks like this (Figure 4):

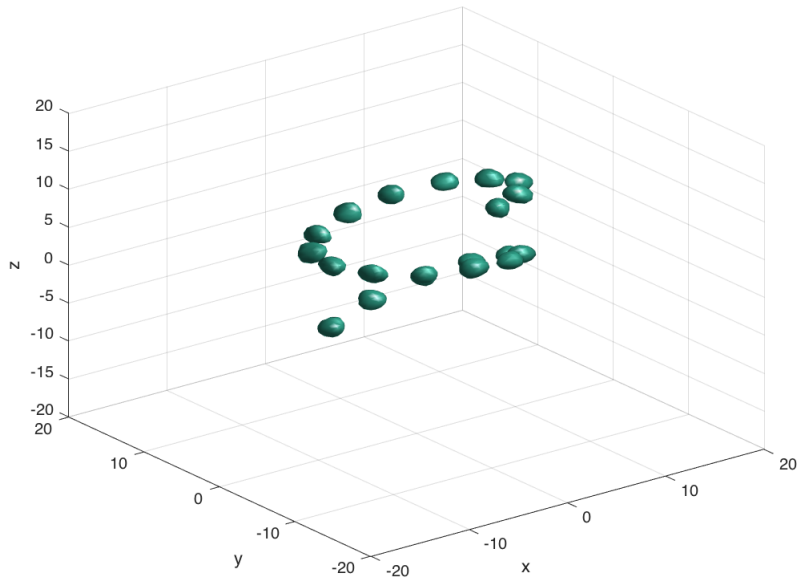

Figure 4: Trajectory of the Marble

Having the trajectory of the marble means that I can save my dog. Then I can find the last location of the marble and ask the doctor use the intense acoustic wave to breakup that marble.

The plot3 from Matlab with label will be like this in Figure 5:

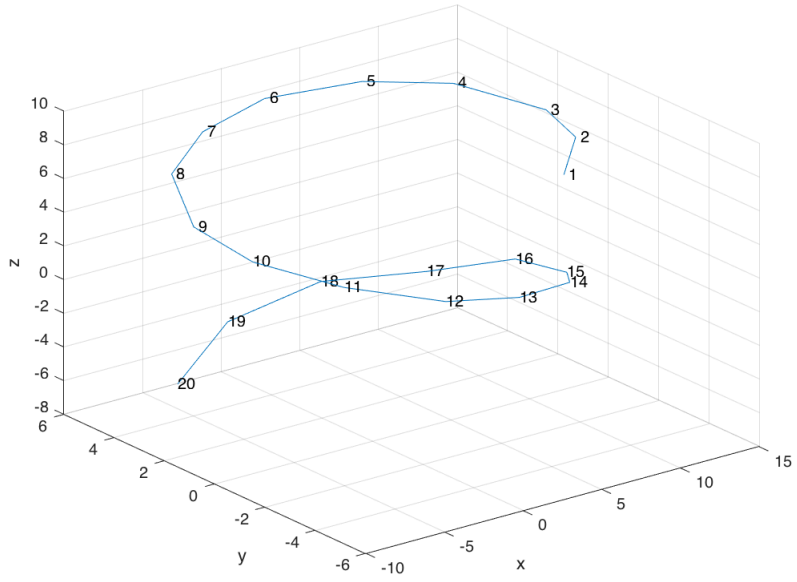

Figure 5: plot3 from matlab

5 SUMMARY AND CONCLUSION

The main idea in this project is applying Fourier transform between spatial domain and frequency domain. In the frequency domain, although the object may be moving all the time, the frequency will stay the same. Also, I use the idea that the average of the sum of all the white noises will approximate to 0 at the end. With these two ideas, I can easily find the center of the frequency and build a Gaussian filter with this center to figure out the useful data in the whole frequency domain. Once I transform the data back to the spatial domain, I can get the location of trajectory I need for my object.

For this problem, I know the final location of the marble in a 3-D coordinate is

$$X = -5.6250 \quad Y = 4.2188 \quad Z = -6.0938$$

The doctor can use this location to break up the marble by using intense acoustic wave.

6 APPENDIX A

MATLAB Functions Used and Brief Implementation Explanations

1. fftn
   Instead of `fft`, I used `fftn`, which would transform all the data by Fourier transfer for 3-D in our case for spatial data.

2. reshape
   Used to reshape the original data to 3-D $64 * 64 * 64$.

3. ind2sub
   Use to find the location indices of the giving number in a matrix or vector.

4. fftshift
   Shifts zero-frequency component to the center of the spectrum.[1]

5. ifftn
   Transfer the frequency data back to the spatial domain with 3-D in our case.

6. isosurface
   Used to plot the 3-D isosurface for our data.

7. plot3
   Plot lines and points in 3-D space.[2]

[1] Copy from Matlab.
[2] Copy from Matlab.

7 MATLAB CODES

```
    clear all; close all; clc
load Testdata.mat
L=15; % 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);

total=0;
for j=1:20
    Un(:,:,:)=reshape(Undata(j,:),n,n,n);
    unt=fftn(Un);
    total=total+unt;
end
ave=abs(fftshift(total))/20;
max1=max(ave(:));
normalize=ave/max1;
[xc,yc,zc] = ind2sub(size(ave),find(ave==max1));
xv=Kx(xc,yc,zc);
yv=Ky(xc,yc,zc);
zv=Kz(xc,yc,zc);
%close all, isosurface(X,Y,Z,abs(Un),0.4)
%xlabel('x');ylabel('y');zlabel('z');
%close all, isosurface(Kx,Ky,Kz,normalize,0.7)
%xlabel('x');ylabel('y');zlabel('z');
%grid on
%axis([-20 20 -20 20 -20 20]), grid on, drawnow
%pause(1)
filter=exp(-0.2*((Kx-xv).^2+(Ky-yv).^2+(Kz-zv).^2));
xyz=zeros(20,3);
for j=1:20
    Un(:,:,:)=reshape(Undata(j,:),n,n,n);
    unt=fftshift(fftn(Un));
    untf=filter.*unt;
    ut=abs(ifftn(untf));
    max2=max(ut(:));
    normalize1=ut/max2;
%       isosurface(X,Y,Z,normalize1,0.7)
%       axis([-20 20 -20 20 -20 20]), grid on
%       drawnow
%       pause(1)
    [xc,yc,zc] = ind2sub(size(ut),find(ut==max2));
    xyz(j,1)=X(xc,yc,zc);
    xyz(j,2)=Y(xc,yc,zc);
    xyz(j,3)=Z(xc,yc,zc);
end
% xlabel('x');ylabel('y');zlabel('z');
% plot3(xyz(:,1),xyz(:,2),xyz(:,3)), grid on
% xlabel('x');ylabel('y');zlabel('z');
% str=cellstr(num2str((1:20)'));
% text(xyz(:,1),xyz(:,2),xyz(:,3),str)
```