AMATH 482 Project1 : An ultrasound problem

Yixuan Liu

January 2020

1 Abstract

In this project, we are given a set of 3d data that records the location of marble in the dog fluffy's intestines in 20 continuous time. However, the data contains huge noises which are generated by the moving of fluffy and the internal fluid movement through the intestines. To eliminate those noises and find the movement of marble, we first average the 20 measurements in the frequency domain and then find the central frequency. Then, we use the Gaussian filter to eliminate noises in the frequency domain. Finally, backing to the time domain, we can find the trajectory without noises.

2 Introduction and Overview

Our dog fluffy swallowed a marble. The vet uses ultrasound to detect the spatial variations in a small area of the intestines to find out where the marble is supposed to be. Due to the movement of fluffy and the internal fluid movement through intestines, we get a set of 3d data with high noises. This data set contains 20 rows of locations of marbles for 20 different measurements in time. In order to save our dog, we need to locate and compute the trajectory of the marble.

First, we apply multidimensional Fourier transform to the dataset, changing the domain from spatial domain to frequency domain. Then, in order to denoise the data, we need to find the center frequency in the data by averaging 20 measurements and looking at the peak frequency. Next, again we apply multidimensional Fourier transform to the dataset, then use Gaussian filter with central frequency we got before to denoise the dataset. After that, we apply inverse Fourier transform to get the denoised location of marble. To find the most possible position of the marble each time, we look at the highest intensity signal for each measurement. Consequently, with this trajectory, we can find the 20th location and the movement of the marble.

3 Theoretical Background

3.1 Fourier Transform

Basic idea of Fourier transform is that taking a signal and breaking it down into different frequencies, or taking a function and breaking it down into sin and cos of different frequencies. Here is the Fourier transform and inverse fourier transform defined on x $\in [-\infty\ \infty]$:

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

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

However, in real life, we only have points, then we choose discrete Fourier transform, assuming we have N points with evenly spaced and x $\in [-\pi\ \pi]$:

$$\widehat{x_k} = \sum_{n=0}^{N-1} x_n e^{2\pi ikn/N} \tag{3}$$

In Matlab, we use fast Fourier transform, which is an improvement of discrete Fourier transform. This FFT assumes working on a 2π period, since our x $\in$ [-L L], we need scale frequency k by $\frac{2\pi}{2L}$. Also, the FFT change the domain of x to x $\in$ [0:L -L:0]. In order to make x $\in$ [-L L], we need use fftshift on the data.

3.2 Gaussian Filtering and averaging noise

We assume that all noises are white noises. To filter out signal, we need know the center frequency. By multiplying the filter, we can magnify frequencies around the center frequency and eliminate other frequencies. Here is the Gaussian filter:

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

where k_0 is the center frequency and τ is the width of filter. In our application, we use 3D filter:

$$F(k) = e^{-\tau(kx-kx_0)^2-\tau(ky-ky_0)^2-\tau(kz-kz_0)^2} \tag{5}$$

In order to find the center frequency, we need a lots of measurements. we need shift those measurements into frequency space, and then average them, and finally find the typical frequency among the average measurement, which is the center frequency. The reason that this averaging works is that we assume noises are in normal distribution with mean 0. If we have enough sample, the average of all sample is around 0.

4 Algorithm Implementation and Development

4.1 Define X,Y,Z

First, we define the interval of x is from -15 to 15. And then, in order to use *FFT* command, we need to discretize x into 64 points. Because x is periodic, we discretize x into 65 points

and choose the first 64 points. We do the same process for y and z. Then, as we mentioned before, the frequency is based on 2π domain. We need to scale it to [-L L].As a result, we multiply k by $2\pi/2L$. Also, we define k from 0 to 2/n and -n/2 to -1, as a consequence of shifting of axis during Fourier transform. However, to make our plot reasonable, we use *fftshift* command to shift k to the domain [-L L]. Finally, we build 3 matrices (64x64x64) for all possible x,y,z value by using *meshgrid* command. Meanwhile, we build another three matrices for kx,ky,kz.

4.2 Averaging measurements

in order to make x,y,z values clear to read, we reshape each row in the data into a matrix(64x64x64). Then we do a multidimensional Fourier transform to each matrix ,and finally averaging the value of 20 matrices to eliminate noises.After that, we do a *fftshift* command on the averaging matrix in order to fit the domain-L to L. Then, to find the center frequency, we first use *max(matrix)* command to find the max value and its index. Then we use *ind2sub* command to gain the index of the specific kx,ky,kz, and plug them into the meshgrid matrix to gain kx,ky,kz, which are center frequencies.

4.3 Filter noises and find trajectory

First, we choose a specific τ value and define the 3d filter.In order to match up with the domain [-L L], we use *fftshift* command to shift the 3d filter. Then, we do the same reshaping process to each row, do a *fftn* command to each reshaped matrix, multiply each matrix with the filter, and transform the clean matrix back to the time domain. This whole process efficiently eliminates most noises. Then we use *max(matrix)* command and *ind2sub* command to find the highest intensity signal for each clean matrix, which tells us the most likely position of marble at each time. The combination of all values that we get gives us the trajectory of marble. Then we use *plot3* command to plot the trajectory.

5 Computational Results

For the center frequency, we have x=1.8850,y=-1.0472 and z = 0. For the trajectory, the marble starts from the star to the dot. The plot is figure 1. For the 20th data measurement, the x,y,z coordinate is (-5.6250,4.2188,-6.0938).

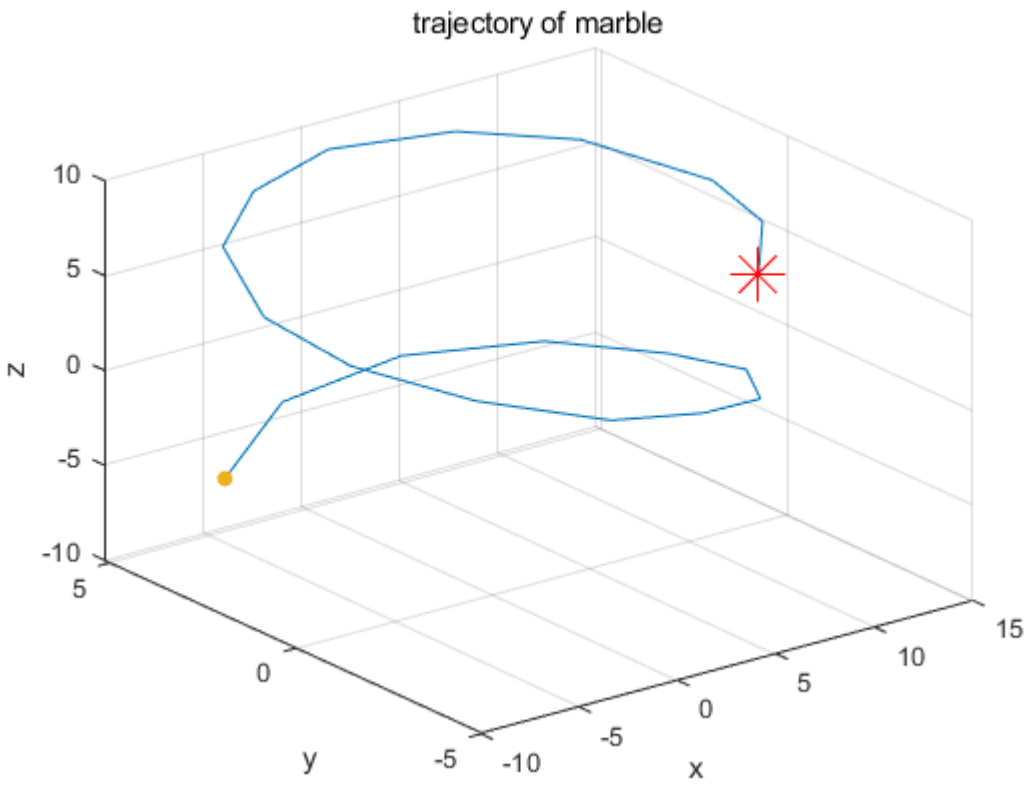

Figure 1: 3D plot of marble's trajectory

6 Conclusion

In this assignment, we use Fourier transform, Gaussian Filter and averaging method to eliminate the noise successfully. we first use averaging method to find out the center frequency, then use Gaussian Filter to eliminate the noise and finally use Fourier transform to change the domain. Also, we figure it out that we should focus an intense acoustic wave on (-5.6250,4.2188,-6.0938) to break up the marble and save dog fluffy.

7 Appendix A: MATLAB functions used and brief implementation explanation

- linspace(x,y,n): generate n points between x and y. we use it to generate x,y,z.
- ffshift(k): Shift zero-frequency component to center of spectrum. we use it to make graph easier to read.
- meshgrid(x,y,z): returns 3-D grid coordinates defined by the vectors x, y, and z. The grid represented by X, Y, and Z has size length(y)-by-length(x)-by-length(z). we use it to discretize x,y,z.

- reshape(A,sz): reshapes A using the size vector. we use it to reshape data in order to clearly read x,y,z data.
- max(frq(:)): get coordinate of maximum value.
- ind2sub(sz,idx): convert linear idx to sz dimension. we use it to find 3d-coordinate of maximum.
- fftn(u): Fourier transform in n-dimansion. we use it to change the domain of data.
- ifftn(u): change back to original domain.
- plot3(x,y,z): plot 3D graph. we use it to plot trajectory of marble.

8 Appendix B: MATLAB codes

```
clear; close all; clc;
load Testdata
%1
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);
uavg=zeros(64,64,64);
for j=1:20
Un(:,:,:)=reshape(Undata(j,:),n,n,n);
utemp = fftn(Un);
uavg=uavg+utemp;
end
frq = fftshift(uavg/20);
[mxv,idx] = max(frq(:));
[kxc,kyc,kzc] = ind2sub(size(frq),idx);
KXC = Kx(kxc,kyc,kzc);
KYC = Ky(kxc,kyc,kzc);
KZC = Kz(kxc,kyc,kzc);

%2
tau = 0.3;
filter = exp(-tau*(Kx-KXC).^2).*exp(-tau*(Ky-KYC).^2).*exp(-tau*(Kz-KZC).^2);
filter = fftshift(filter);
trajectory = zeros(20,3);
```

```
for j=1:20
Un(:,:,:)=reshape(Undata(j,:),n,n,n);
Unt = ifftn(fftn(Un).*filter);
[mxv,idx] = max(Unt(:));
[xt,yt,zt] = ind2sub(size(Unt),idx);
trajectory(j,:) = [X(xt,yt,zt) Y(xt,yt,zt) Z(xt,yt,zt)];
end
plot3(trajectory(:,1),trajectory(:,2),trajectory(:,3));
grid on;hold on;
plot3(trajectory(1,1),trajectory(1,2),trajectory(1,3),'r*','MarkerSize',20)
plot3(trajectory(20,1),trajectory(20,2),trajectory(20,3),'.','MarkerSize',20)
xlabel('x')
ylabel('y')
zlabel('z')
title('trajectory of marble')

%3
x20 = trajectory(20,1);
y20 = trajectory(20,2);
z20 = trajectory(20,3);
```