AMATH 482 HW 1

Yuxin Jiang

January 2020

Abstract

In this assignment, we are trying to locate the trajectory of a marble with data provided through ultrasound taken from a dog's intestine with 20 different measurements that were taken in time. We first averaged the data in frequency domain to find the center frequency, and then utilized Gaussian filter to denoise the data. With the methods mentioned, we can track the path of the marble and then locate the final location so that marble can be removed to save the dog's life.

I. Introduction

Our dog Fluffy accidently swallowed a marble, which we must locate as soon as possible through the ultrasound data to save Fluffy. However, the internal fluid and movement of Fluffy kept generating highly noisy signals. Therefore, we needed to filter the signal using Gaussian filter and determine the location where 20th measurement was taken. In order to remove the noise, we are supposed to take the Fourier transform across the data so that the data is shifted to frequency domain. Through averaging the spectrum, we could then determine the center frequency by taking the maximum value and removing the white noise. We then traced the path of the marble by filtering around the center frequency and converting it back to spatial domain. With denoised data, we will then be able to find the location of the marble with the determination of 20th point in the path.

II. Theoretical Background

Fourier transform is an integral transform defined over the entire line $x \in [-\infty, \infty]$. The Fourier transform and its inverse is defined as,

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

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

where k is the wavenumber. The transform is over the entire real line $x \in [-\infty, \infty]$ whereas our computational domain is only over a finite domain $x \in [-L, L]$. [1]

The Fast Fourier transform (FFT) algorithm is developed by Cooley and Tukey, which could find the transform on the interval $x \in [-L, L]$ with low operation count, high accuracy and periodic boundary conditions on results. It also shifts the data so that $x \in [0,\ L] \rightarrow [-L, 0]$ and $x \in [-L, 0] \rightarrow [0,\ L]$, multiplies every other mode by -1, and assumes a 2π periodic domain.[2] According to these properties, we rescaled the frequencies by $2\pi/L$, where L represents the length of the spatial domain.

Given that white-noise can be modeled by adding a normally distributed random variable with zero mean and unit variance[3], we would be able to remove the white noise by adding and averaging the spectrum so that white noise will add up to zero. While averaging the spectrum, we considered the total number of realizations of incoming signal system, for which white noise is produced for each new realization. The process of averaging could extract a clear frequency signature but could not produce the path in spatial domain.

However, noise attenuation via frequency filtering will help us solve the problem. We used Gaussian filter

$$\mathcal{F}(k) = \exp\left(-\tau(k - k_0)^2\right)$$

Where τ measures the bandwidth of the filter, k is the wavenumber, and k_0 is the center frequency. The generic filter function $\mathcal{F}(k)$ will eliminate the high frequency relative to center frequency and attenuate undesired signals. By converting the filtered data back to spatial domain and tracing the maximum value, we can locate the path of the marble and determine the location by looking at 20th measurement.

III. Algorithm Implementation and Development

We first prepared the data for Fourier transform by discretizing the domain. We set half of the computational domain as 15 and defined the number of Fourier modes as 64 to satisfy the

[1] J.Nathan Kutz, *Data-Driven Modeling & Scientific Computation: Methods for Complex Systems & Big Data*, p. 312
[2] J.Nathan Kutz, *Data-Driven Modeling & Scientific Computation: Methods for Complex Systems & Big Data*, p. 314
[3] J.Nathan Kutz, *Data-Driven Modeling & Scientific Computation: Methods for Complex Systems & Big Data*, p. 325

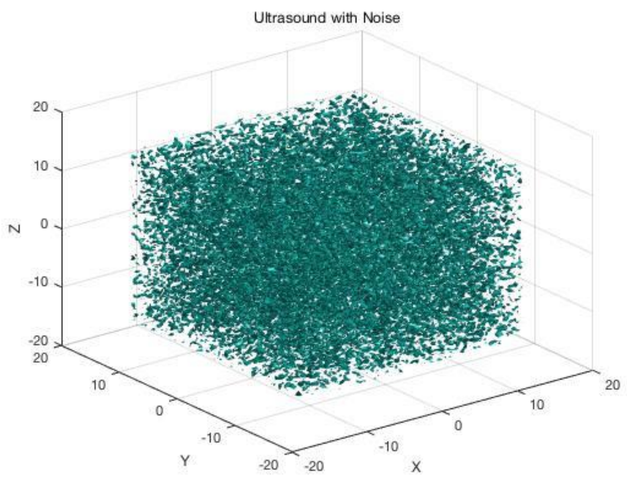
2^n criteria. Due to its periodic boundary condition, we discretized into $n + 1$ points and kept only the first n points due to periodicity. Then we rescaled the frequency by $2\pi/2L$ since FFT assumes working on a 2π periodic domain. Additionally, we also defined the frequency from 0 to $n/2 - 1$ and $-n/2$ to -1 due to the shifting in FFT. In order to visualize the data, we used **meshgrid** to create a grid in both spatial domain and frequency domain. Using the **isosurface**, we plotted the unaveraged ultrasound with noise in spatial domain with reshaped data that is $64 * 64 * 64$. As shown in Figure (1), the data is highly noisy.

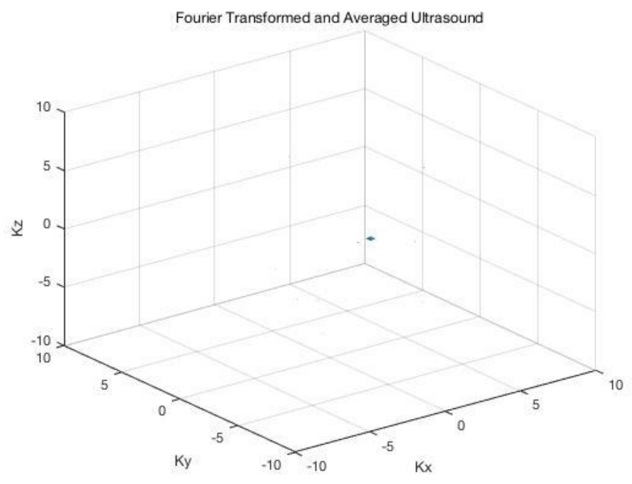
To clean the white noise by averaging the spectrum, we took the Fourier transform by using **fftn** command and determined the average of the sum of 20 sets of data. We used **fftn** instead of **fft** since this is a multidimensional problem, and average of 20 sets of data since this ultrasound was taken in 20 measurements. By using a for-loop, we got the sum and divided it by 20. We used **fftshift** so that the transformed function can be shifted back to mathematically correct position for plotting purpose, which means shifting the zero-frequency to the center. In addition, we normalized the result by taking the absolute value and divided by the maximum value. As shown in Figure (2), we have got a clear picture without white noise after the averaging process. In order to locate the center frequency, we used **max** command to find and locate the maximum value, where N is the index, and **ind2sub** to find the indices in frequency domain. We then plugged indices back in to find the center frequency and corresponding coordinates in spatial domain xfreq, yfreq and zfreq.

Upon finding the center frequency, we could then build a Gaussian filter centering around the center frequency. With a bandwidth of 0.4, we built a 3-D Gaussian filter

$$\mathcal{F}(k) = \exp\left(-\tau((K_x - xfreq)^2 + \left(K_y - yfreq\right)^2 + (K_z - zfreq)^2\right)$$

By multiplying the filter to the Fourier transformed data and inverting it back to the spatial domain, we were able to locate the marble by finding the indices of the maximum value after filtering all the other relatively high frequency. We used **fftshift** twice in this case because we wanted the frequency of the filter and the signal to match up so that we could keep the mathematically correct position while plotting the data. Additionally, in order to trace the path, we needed to put the filtered data of every line in the order of 20 measurements into a matrix so that we could plot it in the graph for visualization, as shown in figure (3). In order to determine the final location, we focused on the 20th measurement and successfully locate the marble by finding indices in spatial domain as shown with a red star in Figure (3).





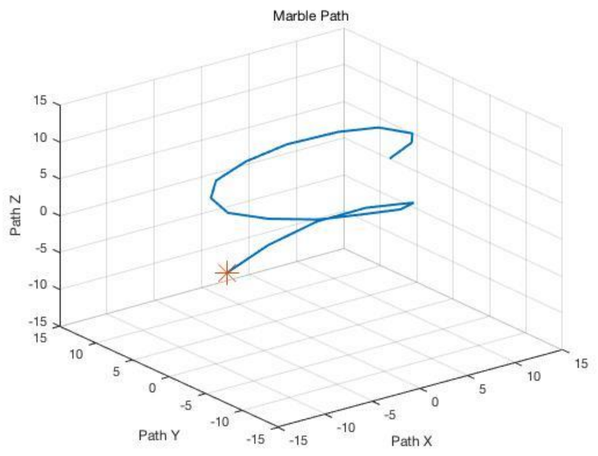


Figure (1): (top left) 3-D isosurface plot of unaveraged ultrasound signal with noise in spatial domain

Figure (2): (top right) 3-D isosurface plot of Fourier transformed and averaged ultrasound signal in frequency domain

Figure (3): (bottom) 3-D isosurface plot of marble path in spatial domain

IV. Computational Results

After averaging the signal, we plotted the center frequency with **isosurface** command with an isovalue of 0.7 since it is hard to see a clear picture of center frequency with isovalue 0.4, which can be shown in Figure (2). Then we were able to locate the frequency signature that

$$xfreq = 1.8850 \quad yfreq = -1.0472 \quad zfreq = 0$$

Then using the filter based on the center frequency, we were able to determine every location of the 20 measurements and using plot3 to plot the path, as shown in Figure (3). The final location of the marble, which is the location in 20th measurement, and it is at

$$x = -5.6250 \quad y = 4.2188 \quad z = -6.0938$$

V. Summary and Conclusions

Although given a highly noisy ultrasound signal, we would be able to utilize averaging method, Gaussian filtering and Fourier transform to find a clear picture of the signal. In our situation, we transformed the data to frequency domain, took an average of Fourier transformed data of different 20 measurements, filtered the data further based on the center frequency found through averaging, and in the end traced the path of the marble. The trajectory of the marble plotted in Figure (3) should successfully show the path of the marble in Fluffy's intestine. The final location of the marble is at $(-5.6250, 4.2188, -6.0938)$ and we suggest focusing on this location so that the marble can be removed and Fluffy can be saved.

Appendix A

MATLAB Function and Implementation

Y = fftn (X) returns the multidimensional Fourier transform of an array using fast Fourier transform algorithm.[4]

Y = fftshift (X) rearranges a Fourier transform by shifting the zero-frequency component to the center or the array.[5]

X = ifftn (Y) returns the multidimensional inverse Fourier transform of an array using fast Fourier transform algorithm.[6]

[X, Y, Z] = ind2sub (sz, N) returns the equivalent multidimensional subscript values translated from index N.[7]

Isosurface (X, Y, Z, F(x), n) creates a 3-D axes with X as x-axis, Y as y-axis and Z as z-axis, plots F(x) in the space with isovalue of n.

[M,N] = max (Y) returns the row indexes of largest elements in each column.[8]

[X, Y, Z] = 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).[9]

plot3 (x, y, ….) plots the set of coordinates in 3-D and connected the points by line.

B = reshape (A,n,n,n) reshapes A into n*n*n array where n represents the size of each dimension.

[4] "Makers of MATLAB and Simulink." MathWorks, www.mathworks.com/.
[5] "Makers of MATLAB and Simulink." MathWorks, www.mathworks.com/.
[6] "Makers of MATLAB and Simulink." MathWorks, www.mathworks.com/.
[7] "Makers of MATLAB and Simulink." MathWorks, www.mathworks.com/.
[8] "Makers of MATLAB and Simulink." MathWorks, www.mathworks.com/.
[9] "Makers of MATLAB and Simulink." MathWorks, www.mathworks.com/.

Appendix B

MATLAB Code

```
% 482 project 1
close all; clear all; clc;
load Testdata
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);
figure(1)
Un(:,:,:)=reshape(Undata(20,:),n,n,n);
isosurface(X,Y,Z,abs(Un), 0.75);
axis([-20 20 -20 20 -20 20]), grid on, drawnow;
xlabel('X'), ylabel('Y'), zlabel('Z');
title('Ultrasound with Noise');

% averaging the spectrum
Utave=zeros(n,n,n);
for j=1:20
Un(:,:,:)=reshape(Undata(j,:),n,n,n);
Utn(:,:,:)=fftn(Un);
Utave=Utave+Utn;
end
Utave=abs(fftshift(Utave)/20);
Utave=Utave/max(Utave(:));%normalize

figure(2)
```

```
isosurface(Kx,Ky,Kz,Utave, 0.7);
axis([-10 10 -10 10 -10 10]), grid on, drawnow;
xlabel("Kx"), ylabel("Ky"), zlabel("Kz");
title("Fourier Transformed and Averaged Ultrasound");

%find central frequency
[M,N]=max(Utave(:));
[Kxx,Kyy,Kzz]=ind2sub([n,n,n],N);
xfreq=Kx(Kxx,Kyy,Kzz);
yfreq=Ky(Kxx,Kyy,Kzz);
zfreq=Kz(Kxx,Kyy,Kzz);

%filtering and find path
tau=0.4;
filter=exp(-tau*((Kx-xfreq).^2+(Ky-yfreq).^2+(Kz-zfreq).^2));
path=zeros(20,3);
for j=1:20
    Un(:,:,:)=reshape(Undata(j,:),n,n,n);
    Utn=fftshift(fftn(Un));
    Unft=fftshift(filter.*Utn);
    Unf=ifftn(Unft);
    [Q,W]=max(Unf(:));
    [px,py,pz]=ind2sub([n,n,n],W);
    xpath=X(px,py,pz);
    ypath=Y(px,py,pz);
    zpath=Z(px,py,pz);
    path(j,1)=xpath;
    path(j,2)=ypath;
    path(j,3)=zpath;
end

figure(3)
plot3(path(:,1),path(:,2),path(:,3),'LineWidth', 2)
axis([-15 15 -15 15 -15 15]), grid on, drawnow
```

```
xlabel('Path X'), ylabel('Path Y'), zlabel('Path Z')
title('Marble Path');
hold on

%find the final place
plot3(path(20,1),path(20,2),path(20,3),'*','MarkerSize',20);
location=[path(20,1) path(20,2) path(20,3)];
```