Amath 482 Homework 1: An Ultrasound Problem

Rose Ju

January 2020

Abstract

We were given a noisy ultrasound of our dog Fluffy's intestines due to the suspicion that he has swallowed a marble. We were hoping to locate and identify the trajectory of the marble through the provided 20 data measurements taken in time. The methods to find the marble is to first average the spectrum and determine the central frequency. Then, we will apply a Gaussian filter around these frequencies to denoise the data and find the path of the marble. Lastly, we will locate the marble with the denoised data.

I. Introduction and Overview

Our dog Fluffy has swallowed a marble and it has traveled down to the intestines due to the vet's suspicion. However, we do not know the location nor the trajectory of the marble. Using ultrasound, we were able to obtain a dataset that contains 20 rows of data for 20 different measurements takes at the time. Unfortunately, due to the intense movements and the internal fluid movements of Fluffy, we were only able to obtain highly noisy data. In order to save our dog, we must determine the location of the marble at the 20_{th} data measurement and use an intense acoustic wave to break up the object.

First, we must remove the excessive noises from the data by averaging the spectrum and determine the central frequency generated by the marble. In order to do that, we will apply a Fourier transform of the 3-D ultrasound data and take the average of the results at every time points. Then we can

find the central frequency. After finding the frequency, we will apply a spectral filter around the central frequency which will remove the unnecessary noises in the dataset. Once our data has been denoised, we will inverse the Fourier transform. Finally, we will be able to determine the path of the marble and find its location at the 20th measurement.

II. Theoretical Background

i. Fourier Transform

The Fourier Series is a summation of series sines and cosines that is used to formulate a function:

$$f(x) = \frac{a_0}{2} + \sum_{n=1}^{\infty} (a_n \cos(nx) + b_n \sin(nx)) \quad x \in (-\pi, \pi]$$

This expansion is a 2π periodic function because it is formed by sines and cosines on the interval $(-\pi, \pi]$.

Relating to the Fourier Series, we have the Fourier Transforms, which was used in this project. The Fourier Transform is an integral transformation of a time function into its frequencies.

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

where k is wavenumber. If we inverse of the fourier transform, then we get the inverse fourier transform as:

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

The algorithm implemented in this project is an improved version of the fourier transform called, Fast Fourier Transform. It performs the fourier transform and its inverse at a much faster time from its original time complexity $O(N^2)$ to $O(NlogN)$. This upgraded version of the algorithm have faster speed and provides high accuracy. In addition, it shifts the data from $x \in [-L, 0]$ to $[0, L]$ and $x \in [0, L]$ to $[-L, 0]$ and, by default, it assumes the function is on a 2π periodic domain. Therefore, in order to better visualize the data, we must use `fftshift` to undo the shifts ($\frac{2\pi}{L}$, where L is the length of the finite domain) in FFT.

ii. Time Averaging

The noises in the data can be modeled by adding normal distribution random variable with 0 mean and a finite variance. By averaging all the noises over signals, the noises should add up to

0 and therefore, remove the unnecessary noises in the data. To achieve this process, we need to first transform the data using `fft()` and rearrange the data so the 0 frequency component is at the center. Then, we will take the its absolute value and divide by the maximum to find the normalized data. Finally, by searching for the maximum value, we will be able to find the central frequency.

iii. Spectral Filtering

Highly noisy data is often an issue for analyzing datasets, but they can often be solved by filtering the data with a well-chosen filter function. Spectral filtering is a method to be focused on an interested frequency, which removes the unwanted frequencies from the signal and better determine the interested frequency in the noisy data. In this project, we will be using the Gaussian filter:

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

where $k$is the wavenumber, k_0 is the central frequency and τ is the bandwidth of the filter. Using this filter, we will be able to denoise the unwanted frequencies from the central frequency. We will be multiplying this filter to the fourier transformed data and removes the uninterested frequencies. Then we will use the inverse fourier transform to get the results back into the time domain.

III. Algorithm Implementation and Development

- Load `Testdata.mat`
- Discretize the domain
  - We should discretize into n+1 points and take the first n points
- Define wavenumber k
  - Set k to be 0 to $\frac{n}{2} - 1$ and $-\frac{n}{2}$ to 1 due to the shifting of the fourier transform
  - Rescale k by 2π/L because `fft` is 2π periodic signals.
- Use `fft()` to shift k
  - Rearrange the frequency using `fftshift()`, so the 0-frequency component is at the center
- Create the meshgrid using the shifted frequencies

- Take each row of the data and reshape into 3D and take it's Fourier transform using `fftn()`
- Add up each iteration and take its average
    - Normalize the data and take the absolute value
- Determine the central frequency by finding the maximum value in the matrix
- Set up Gaussian filter by entering the discovered central frequency
    - Multiply the data with the Gaussian filter to denoise
- Use `ifftn()` to invert the FFT and covert back to spatial domain
- Find the maximum values in the matrix and determine the path of the marbles
- Run through all iterations and stores final location

IV. Computational Results

After we normalized the transformed average data, we found our central frequency to be (1.8850,-1.0472,0) with respect to (x, y, z). Figure1 is a 3D graph of the normalized fourier transformed data in the frequency domain.

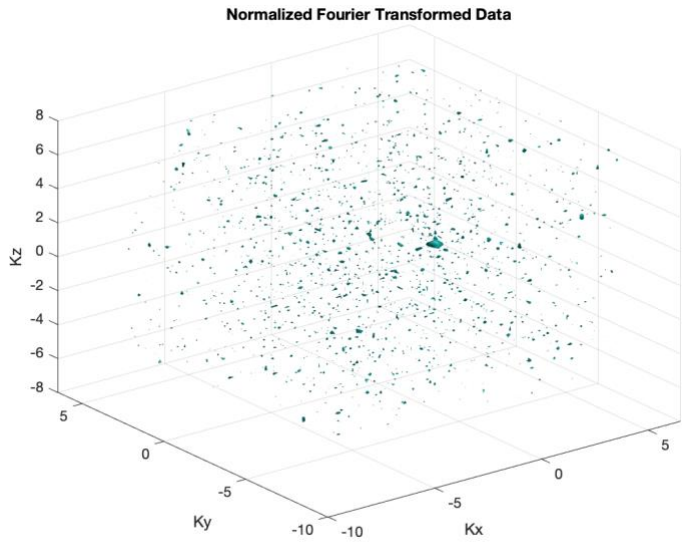

Figure 1: 3-D isosurface plot in the frequency domain of the normalized Fourier Transformed Data

Then we applied our Gaussian filter with the bandwidth of the filter τ as 0 and plugged in the central frequency found above, we were able to denoise our data. After multiplying the `ifftn()` of the filter to all 20 measurements of the data, we were able to find the location of the marble at the 20th point, which is (-4.8171,5.6549,-6.7021) with respect to (x, y, z). Figure2 is a 3D illustration of the path of the marble traveling down the intestines.

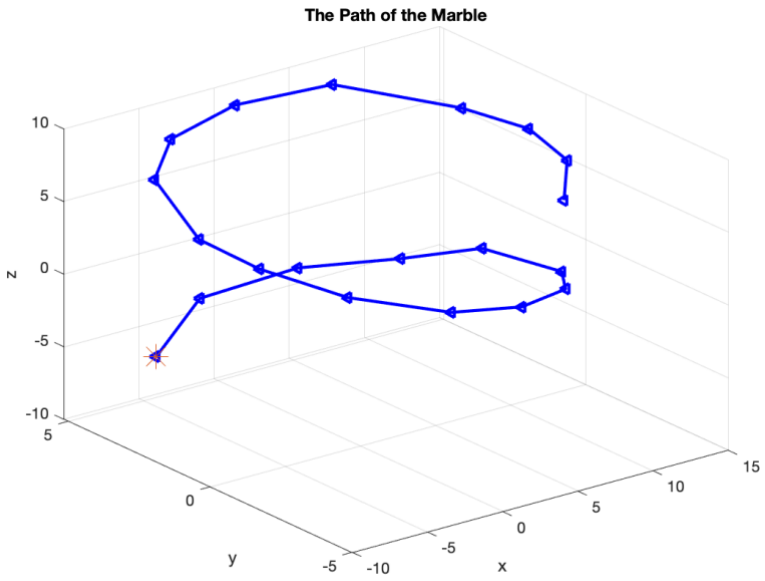

Figure 2: 3-D plot3 visualization the travel path of marble

V. Summary and Conclusions

We were given a noisy ultrasound of our dog intestines and were trying to identify the location of a swallowed marble within the intestines. First, we took the Fourier transform of the data set and transformed the data from spatial domain into frequency domain. Then, we located the central frequency at (1.8850,-1.0472,0) by taking the average of the 20 measurements of the transformed dataset. Afterwards, we applied a Gaussian filter with the bandwidth τas 0.1 to the data. With the denoised data, we were able to estimate the path of the marble and located the marble location at the 20_{th} measurement, which turned out to be at (-4.8171,5.6549,-6.7021).

Appendix A – MATLAB functions

`abs(X)` – calculates the absolute value of the input element. If input in complex, then returns the complex magnitude. This was used to normalize our dataset.

`fftn(X)` – performs fast fourier transform to the given data. We used this to transform our data set into frequency domain.

`fftshift(X)` – rearranges the elements of X and place 0-frequency to the center of the vector. We used this to translate our data for better visualization.

`ifftn(X)` – performs inverse fast fourier transform to the given data. We used this to reverse the `fftn(X)` transformation back to spatial domain.

`ifftshift(X)` –rearranges the elements of X and place 0-frequency to the center of the vector. We used this to translate our data for better visualization.

`ind2sub([],i)` – given index and transforms the index into indices of the given matrix. This was used to determine the central frequency after finding the maximum.

`linspace(X1,X2,n)` – creates a vector of `n` evenly spread elements from `X1` to `X2`. This was used to create linearly spaced vector for the spatial domain.

`load` – load data from MAT-file into the tensor. We used this to load the ultrasound dataset into our workspace.

`max(X)` – calculates the maximum value of the input element. The function was used to calculate the normalization of the spectrum.

`meshgrid(X,Y,Z)` – creates a 3-D grid with the given vectors. We used this to create the spatial and frequency domain.

`reshape(A,[M,N])` – reshapes the vector `A` to the given matrix sizes. Used to calculate the average of each iteration

`zeros(X,Y,Z)` – creates a matrix of zeros `X * Y * Z`. The function was used to initialize empty matrixes with predetermined size.

Appendix B – MATLAB codes

```
clear all; close all; clc;
load Testdata

%Basic set up
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);

%fftn shift
avg = zeros(n,n,n);
for i = 1:20
    avg = avg + fftn(reshape(Undata(i,:),n,n,n));
end

%Normalize
avg = abs(fftshift(avg)) ./ max(abs(avg(:)));
figure(1)
isosurface(Kx,Ky,Kz,fftshift(avg),0.5)
grid on, drawnow
xlabel("Kx"), ylabel("Ky"), zlabel("Kz")
title("Normalized Fourier Transformed Data");

%Center frequency
[M, index] = max(avg(:));
[Xi, Yi, Zi] = ind2sub([n,n,n],index);
xc = ks(Yi);
yc = ks(Xi);
zc = ks(Zi);

%Gaussian Filter
bw = 0.1;
filter = exp(-bw * (Kx - xc).^2 + -bw * (Ky - yc).^2 + -bw * (Kz - zc).^2);
filter = fftshift(filter);

%Find Path
path = zeros(20,3);
for i = 1:20
    Unds(:,:,:) = reshape(Undata(i,:),n,n,n);
    dsf = filter.*fftn(Unds);
    dsf = ifftn(dsf);
    [M, i2] = max(abs(dsf(:)));
    [xp,yp,zp] = ind2sub([n,n,n], i2);
    path(i,1) = X(xp,yp,zp);
    path(i,2) = Y(xp,yp,zp);
    path(i,3) = Z(xp,yp,zp);
end

plot3(path(:,1),path(:,2),path(:,3),'b-.','LineWidth',2);
grid on, drawnow
```

```
title('The Path of the Marble');
xlabel("x"), ylabel("y"), zlabel("z")
hold on;
plot3(path(20,1),path(20,2),path(20,3),'*','MarkerSize',15);
```