AMATH 582 Homework 1

Shannon Dow

January 24, 2020

Abstract

The following report describes an algorithm that takes three-dimensional signal data from an Ultrasound and uses it to find the path of a marble through the inside of a dog. Mathematically, this will involve Fourier and inverse Fourier transforms, in order to process frequency data about the marble. Fourier transforms are incredibly powerful due to their orthogonality in function space, and provide the tools to analyze the data in different domains. The algorithm first finds the frequency signature of the marble by averaging the signals in the Fourier domain. It then constructs a filter around that frequency and applies it to the data in order to de-noise. Finally, using inverse Fourier transforms, the algorithm finds where center frequency occurs in the spacial domain at each time instance to track the marble's location with time. With this information, we can rescue the dog by safely destroying the marble.

1 Introduction and Overview

1.1 The Problem

In this assignment, a dog has swallowed a marble that is now moving through its intestines. Ultrasound is providing signal data as to the location of the marble, however, since the dog is moving, it disrupts the signals of the ultrasound creating noisy data. In order to locate the marble's final location and neutralize the threat, I must find the trajectory of the marble through the dog. The data from the ultrasound contains 20 realizations of three dimensional signal data relating to the location of the marble. Each of the 20 realizations are at different time increments. To perform analysis on and process this data, it will be useful to use the Fourier transform and inverse Fourier transform.

1.2 Solution Overview

To begin, I will find the frequency signature generated by the marble. Since the Fourier transform expands in the frequency domain, I will transform the data and then average to find the center frequency of the marble. The center frequency of the marble should be the maximum of these, since the ultrasound sends waves that should bounce off the marble and not the surrounding tissue. Then, in order to de-noise completely, I will construct a filter around that center frequency and determine the path of the marble using the inverse Fourier transform. Finally, with the path determined, I will find the final location of the marble in the intestines so that we can send an intense acoustic wave to break up the marble and save the dog.

2 Theoretical Background

For this assignment, I will be using the Fourier transform to perform analysis on the data. The Fourier transform , shown below, can be used to represent any function as sin and cos functions. Expanding our signal data in the Fourier (or frequency) domain is key to this assignment, since we need to analyze the frequency of the signal data provided by the Ultrasound. The a_n and b_n represent "how much" to go in each of the cos(nx) and sin(nx) directions in function space. The Fourier transform is orthogonal since the inner product is zero in function space. With orthogonality, we can represent any function uniquely using the Fourier transform.

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

The Fourier transform can also be expressed in terms of a complex exponential where k is the frequency.

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

This means that the data in the Fourier domain will have complex components. When potting, it is important to only graph the absolute value. Additionally, the Fourier transform expects data that is between $-\pi$ and π, so it is important to scale the domain of k so it has length 2π. The Fourier transform gives you frequency information, but not when the frequencies occur. Thus, once the frequency signature is determined, we want to get information back in the spacial domain. For this, we use the inverse Fourier transform.

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

2.1 Finding the Center Frequency:

The data provided in this assignment is three dimensional signal data about a marble moving through a dog. Since the dog is moving and shifting the marble in the internal fluid, there is noise on the signal data. Additionally, since the marble is moving in time, we cannot average the signals in the time domain, as that would not give us relevant information. The signal frequency of the marble is fixed, so instead, we can average in the Fourier domain. To do this, we expand using the Fourier transform. Averaging the signal de-noises the data, since if we assume that the noise is distributed around the frequency, the noise will go to zero. It is important that you do not take the absolute value of the data until after averaging, since expanding in the Fourier domain will result in complex numbers. After the data is averaged, we want to calculate the maximum of the frequencies, as this will give the best approximation to the frequency signature of the marble. Once we find where the maximum frequency occurs, we can construct a filter centered around it.

2.2 Filtering the Data:

While we can technically use any number of filters, in this assignment, we will use the Gaussian filter, which is fairly standard and the smoothness results nice properties. The data is in three dimensions, so the three dimensional Gaussian Filter will be used:

$$filter = exp(-a * [(KX - kx)^2 + (KY - ky)^2 + (KZ - kz)^2])$$

The "a" is a parameter for the width of the Gaussian filter. This can be adjusted so we get the best results. Depending on the size of the filter, it will get better resolution on either low or high frequencies. A large filter picks up low frequencies and a small filter picks up high frequencies. We want to apply the filter to all instances of the transformed data, or data in the frequency domain. Once we multiply the transformed data by the filter around the maximum frequency, we can inverse Fourier transform it to get back into the spacial domain. This allows us to track the location in time. To find the path of the marble, we want to find the location of the maximum of the filtered data at all twenty instances. This will give us 20 (x,y,z) coordinates of the marble we can use to visualize its trajectory within the dog. The last coordinate will be the final location of the marble, and will therefore be where we will want to send the intense acoustic wave to break up the marble.

3 Algorithm Implementation and Development

3.1 The Data and Original Setup

The data provided in this assignment is 20x262144. It contains three dimensional signal data at 20 different times, or realizations. To keep track of transforms, shifts, and filters throughout the code, the following letters will be appended to classify Un.

1. t for transform

2. s for shift

3. f for filter

Algorithm 1: Importing Data and Setting Up Spacial and Frequency Domains

```
L = 15;
n = 64;
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);
```

3.2 Algorithm 1

The first two lines establish the spacial domain (L) and the number of Fourier modes (64). It is important for the Fourier Transform that the number of Fourier modes be 2^n. Although the Matlab Fourier transform function will run without it, it makes it more efficient. Next, the time domain is discretized. We will have x, y and z signal data at each time. The spacial domain is then scaled down so it only has length 2π since the Fourier transform function expects 2π periodic domain. The Fourier domain is then shifted using `fftshift`. It is shifted so that when plotting, it is easier to visualize the frequencies. Finally, using the Matlab `meshgrid`, functions, six different cubes of data are created, 3 for the time domain, and three for the frequency domain. We can now find the signal data based on time or frequency.

Algorithm 2: Finding the Marble's Frequency Signature

```
avg = zeros(n,n,n);
for j = 1 : 20 do
  Un(:,:,:)=reshape(Undata(j,:),n,n,n);
  Unt =fftn(Un);
  avg = avg + Unt;
end for
avg = fftshift(avg)/20;
[maxValue,index] = max(abs(avg(:)));
[i1,i2,i3] = ind2sub(size(avg),index);
kxo = Kx(i1,i2,i3); kyo = Ky(i1,i2,i3); kzo = Kz(i1,i2,i3);
```

3.3 Algorithm 2

The second section of the code finds the marble's frequency signature. First, I initialize a variable to store the average of the transformed data. Then, for each instance of the 20 times, the data is first reshaped. This is done to put it into a format that is easier to work with, since it converts it into 64x64x64 cubes, which is based on the number of Fourier modes. Then, since the marble is not stationary and we want to find the frequency signature, we must use `fftn` to expand in the Fourier domain. Using the loop, the average is updated each iteration. After the loop, you take `fftshift` of the average and divide it by all 20, the number of realizations. Since the Fourier domain at the start was shifted, we also need to shift the average so that coordinates match. Now that the frequency data was partially de-noised, we want to find the maximum of the absolute value of the average frequencies. This will give us the frequency signature of the marble. The `max` function in Matlab provides the maximum value and the index of the average. We want the linear indices of the maximum frequency so we can find the corresponding frequency (k values) from the meshgrid. For this, I make use of the Matlab `ind2sub` function. Now that we have the 3 numbers that correspond to the

frequency signature in each direction, we can construct a filter. To center the filter around that frequency, set:

$$filter = e^{\frac{-a}{2}((Kx-kxo)^2+(Ky-kyo)^2+(Kz-kzo)^2))}$$

Algorithm 3: Filtering the Data and Finding the Path of the Marble

```
xvec = []; yvec = []; zvec = [];
for j = 1 : 20 do
  Un(:,:,:)=reshape(Undata(j,:),n,n,n);
  Unt =fftn(Un);
  Unts = fftshift(Unt);
  Untsf = Unts.*filter;
  Untf= ifftshift(Untsf);
  Unf = ifftn(Untf);
  [maxValue,index2] = max(abs(Unf(:)));
end for
```

3.4 Algorithm 3

First, three vectors are initialized. This will be to keep tack of the location of the marble's position at each of the 20 time intervals. Inside the for loop, the data is again reshaped, as above, and transformed into the Fourier domain. Since we shifted our frequency k values at the beginning, it is important to shift the transformed data before filtering. Next, multiply the filter by the shifted data. This takes information inside the filter and makes everything else zero. Then, before using `ifftn` which is the inverse Fourier transform, the data must be shifted back, since `ifftn` anticipates unsifted data. This has allowed us to track the marbles signature frequency in the frequency domain and then taken that information back to the spacial domain so we know when and where the marble is traveling. similarly to above, we take the maximum of the absolute value to get the best estimate of the marble's position at each time. Then, again using the `ind2sub` function in Matlab, it is possible to find the linear index of the these positions, so we can find them in the meshgrid.

4 Computational Results

4.1 Question 1

First, I found the frequency signature of the marble, through transforming data into the Fourier domain and averaging the spectrum. Using the `ind2sub` Matlab function, I was able to determine the center frequency to be: (1.8850,-1.0472,0). This was where the maximum of the average was located.

4.2 Question 2

Using the results from question 1, I centered the gaussian filter around the center frequency of the marble and then transformed the data back into the spacial domain to track it's path

4.3 Question 3

Once the data has been filtered and de-noised, to find the location of the marble at each time instance, I found the maximum of the filtered data in the spacial domain. Using the `ind2sub` Matlab function, I was able to get the subscripts of the position to be able to make a plot of the trajectory. The final location of the marble is the position at the final (20th) instance in time: (-5.6250, 4.2188, -6.0938)

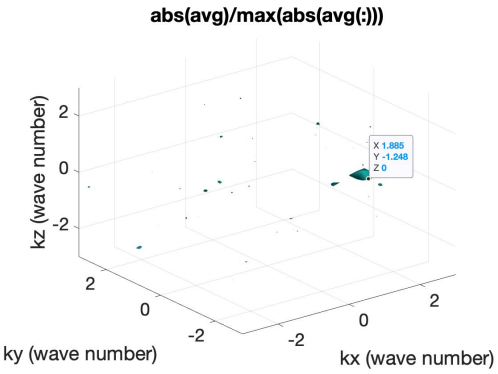

Figure 1: This is plotting the average frequency divided by the maximum of the average frequency in the Fourier domain

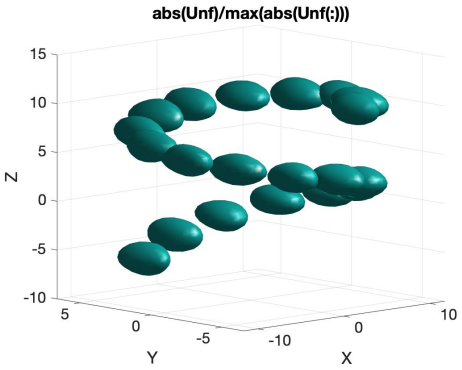

Figure 2: This is plotting the filtered data divided by the maximum of the filtered data in the spacial domain

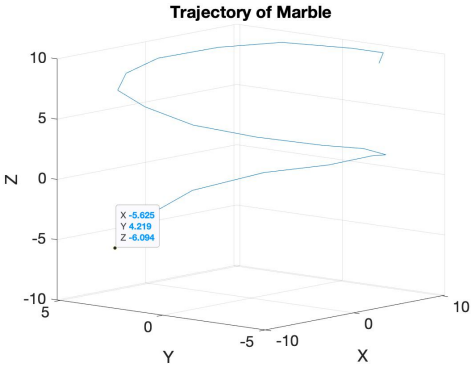

Figure 3: This is plotting the trajectory of the marble in three dimensions

5 Summary and Conclusions

To summarize, this algorithm takes three dimensional signal data from an Ultrasound and finds the path of marble through a dog's intestinal track. Using Fourier Transforms, the algorithm averages the signal data in the frequency domain to find the central frequency of the marble. It then uses that to center a Gaussian filter to de-noise the signal data. Using inverse Fourier transforms, the algorithm tracks and plots the marble in the spacial domain and finds the final position of the marble. Since the marble is moving in time, changing the domain to and from the spacial and frequency domains is key to understanding both the frequency and spacial components of the data. This allows us to find an accurate location of the marble to save the dog's life.

Appendix A MATLAB Functions

- `y = linspace(-L,L,n+1)` returns a row vector of `n+1` evenly spaced points between `-L` and `L`.
- `[X,Y,Z] = meshgrid(x,y,z)` returns 3-D grid coordinates based on the coordinates contained in the vectors `x` and `y` and`z`. X,Y, and Z are then cubes of data formed from these vectors.
- `reshape()` Since Undata starts out as 20x262144, `reshape()` turns this into a 64x64x64 data object so that we have x, y, and z signal data together at each time instance.
- `fftn()` This is the n-dimensional fast Fourier transform of the data
- `fftshift()` This shifts the domain of the Fourier transform so that it is more readable.
- `[maxValue,index] = max()` The max function finds both the maximum value, and the linear index of that maximum value.
- `[i1,i2,i3] = ind2sub(size(),index);` The ind2sub function takes in the size of an object and the single linear index and returns the corresponding subscripts, so the x, y, and z at that linear index.
- `ifftshift` This is the inverse Fourier transform shift. This takes the shifted data and returns it to the original unshifted Fourier domain.
- `ifftn` This is the inverse n-dimensional Fourier transform. It takes the transformed data and returns it to the original spacial domain.

Appendix B MATLAB Code

B.1 Github Link

https://github.com/shannondow/AMATH-582-Homeworks-2020

```
clear all; close 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;
%Scale the fourier domain, since it is expecting a periodic domain of 2pi
k=(2*pi/(2*L))*[0:(n/2-1) -n/2:-1]; ks=fftshift(k);
[X,Y,Z]=meshgrid(x,y,z);
%Using the shifted ks values in meshgrid
[Kx,Ky,Kz]=meshgrid(ks,ks,ks);
%Initialize the average
avg = zeros(n,n,n);
for j=1:20
    %The reshape is used to make data into a 20 realizations of a cube of
```

```
    %data
    Un(:,:,:)=reshape(Undata(j,:),n,n,n);
    %Find the Fourier Transform of the data:
    Unt =fftn(Un);
    %Average in the Fourier Domain
    avg = avg + Unt;
end


%Find the average:
avg = fftshift(avg)/20;
%Plotting the normalized average, partially denoised data
%The isosurface constant is used to scale the image
figure(1)
close all,isosurface(Kx,Ky,Kz,abs(avg)/max(abs(avg(:))),0.6)
axis([-3 3 -3 3 -3 3]), grid on, drawnow
xlabel('kx (wave number)');
ylabel('ky (wave number)');
zlabel('kz (wave number)');
title('abs(avg)/max(abs(avg(:)))')
%Find the maximum frequency
[maxValue,index] = max(abs(avg(:)));
%Find linear index the maximum:
[i1,i2,i3] = ind2sub(size(avg),index);

kxo = Kx(i1,i2,i3);
kyo = Ky(i1,i2,i3);
kzo = Kz(i1,i2,i3);

%Create a Filter:
a = 1; %Width of Filter
filter=exp(-0.5*a*((Kx-kxo).^2+(Ky-kyo).^2+(Kz-kzo).^2));
%isosurface(Kx,Ky,Kz,abs(filter),0.5)

%Initialize vectors to hold data about the location of the dog
xvec = [];
yvec = [];
zvec = [];
for j= 1:20
    Un(:,:,:)=reshape(Undata(j,:),n,n,n);
    %Transform the data
    Unt = fftn(Un);
    %Shift that data, since the meshgrid uses shifted K values, so should
    %our filter in the Fourier domain
    Unts = fftshift(Unt);
    %Apply the filter
    Untsf = Unts.*filter;
    %Unshift the data since the inverse Fourier transform expects unshifted
    %data
    Untf= ifftshift(Untsf);
    %Apply the inverse Fourier transform to get back into the time domain
    Unf = ifftn(Untf);

    %Plots the path of the filtered data in the time domain
```

```
69      figure(2)
70      isosurface(X,Y,Z, abs(Unf)/max(abs(Unf(:))), 0.6)
71      xlabel('X')
72      ylabel('Y')
73      zlabel('Z')
74      title('abs(Unf)/max(abs(Unf(:)))')
75      [maxValue,index2] = max(abs(Unf(:)));
76      grid on;
77
78      %Finds the maximum in the filtered data, and then the linear index
79      [i1,i2,i3] = ind2sub(size(Unf),index2);
80      xnew = X(i1,i2,i3);
81      ynew = Y(i1,i2,i3);
82      znew = Z(i1,i2,i3);
83
84      %Adds the location (x,y,z) into respective vectors
85      xvec = [xvec, xnew];
86      yvec = [yvec ,ynew];
87      zvec = [zvec, znew];
88  end
89  %Plots the trajectory of the marble
90  figure(3)
91  plot3(xvec,yvec,zvec);
92  xlabel('X')
93  ylabel('Y')
94  zlabel('Z')
95  title('Trajectory of Marble')
96  grid on;
97
98  %Final location of marble at the twentieth instance
99  disp(xvec(20));
100 disp(yvec(20));
101 disp(zvec(20));
```