AMATH 582 Homework 1

Christopher Liu

January 24, 2020

Abstract

My dog Fluffy has swallowed a marble and I will need to have it removed in order to save Fluffy's life. 20 ultrasound measurements taken in time with high levels of noise were made to determine the location of the marble. By applying concepts from Fourier transforms to analyze the data in the frequency domain, we are able to de-noise the data by employing averaging and filtering and ultimately determine the location of the marble.

1 Introduction and Overview

Fluffy the dog has swallowed a marble and will need to undergo life-saving treatment to remove the marble. It is believed that the marble is currently in Fluffy's intestines and an ultrasound scan has provided us with data concerning the spatial variations in a small area the marble is suspected to be located. In total, 20 different measurements were taken in time of Fluffy's intestine. However, the data has high levels of noise due to internal fluid movement and because Fluffy was moving during the scan.

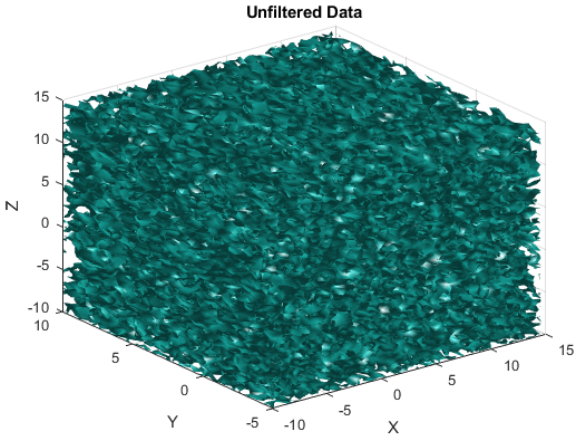

Figure 1: Isosurface of the 20th measurement

Plotting the 20th measurement as an isosurface, we can observe how difficult it is to pick out a signal from all the noise (See Figure 1) . In order to determine the location of the marble so that it can be broken up using an intense acoustic wave, we have been tasked with applying mathematical methods to de-noise the data. We will analyze the ultrasound data in the frequency domain through the use of Fourier transforms and determine the position of the marble in each measurement by employing averaging and filtering in order to save Fluffy.

2 Theoretical Background

To de-noise the data, we will employ Fourier transforms to analyze the data in the frequency domain. In doing so, we will be able to determine the marble's frequency signature using averaging and then create a

filter around the center frequency to determine the marble's position.

2.1 Fourier Series and Transforms

Fourier introduced the concept of representing a given function $f(x)$ by trigonometric series expansion of sines and cosines[1]:

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

However, it is important to note this expansion produces 2π-periodic solutions. Extending this concept to the entire line (from $x \in (-\pi, \pi]$ to $x \in [-\infty, \infty]$), we get what is known as the Fourier transform (2) and its inverse, where k are the continuous wavenumbers (Equation 3).

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

$$f(x) = \frac{1}{\sqrt{2\pi}} \int_{-\infty}^{\infty} e^{ikx} F(k) dk \quad x \in [-\infty, \infty]. \tag{3}$$

Since we do not want to be restricted to 2π-periodic solutions, we will use the Fourier transform to move from the spatial domain to the frequency domain. Working in the frequency domain is advantageous because the frequency signature of the marble does not change with time, unlike its spatial position. Once the frequency signature has been identified, we will then determine the spatial position by performing the inverse operation (3).

2.2 Averaging

Averaging the measurements taken in time in the frequency domain allows us to remove much of the white noise in the data. This is because white noise has a *zero mean*, which means that noise should in theory sum to 0[1].With this in mind, averaging will allow us to determine the center frequency with more certainty as the center frequency should be unaffected by averaging. Furthermore, since the center frequency of the marble does not change with time, we do not need to worry about the marble moving with time in Fluffy's intestines.

2.3 Spectral Filtering

Once the center frequency has been determined, spectral filtering can be applied to extract information from a specific frequency. There are many different types of filters that can be applied to this problem[1]. For this problem, we have chosen to use a Gaussian as it is simple.

$$\text{Gaussian Filter in 3D} = e^{-\tau((K_x - K_{x_0})^2 + (K_y - K_{y_0})^2 + (K_z - K_{z_0})^2} \tag{4}$$

where τ is the width or bandwidth of the filter, (K_x, K_y, K_z) are the wave numbers and $(K_{x_0}, K_{y_0}, K_{z_0})$ is the center frequency at which the filter is centered at. Through filtering, we will be able to remove the noise as well as undesired frequencies.

3 Algorithm Implementation and Development

Before the dataset is manipulated, we first need to rescale our wave numbers over the domain as the `fft` assumes that the signals are 2π periodic. This is accomplished by multiplying each wavenumber by $\frac{2\pi}{2L}$. Also to create the filter and for plotting, we will create a meshgrid of the shifted wavenumbers by using `fftshift` and `meshgrid` on our rescaled wavenumbers. A `meshgrid` for the spatial domain is also created in (x, y, z) for plotting purposes.

Two main steps were taken to find the final position of the marble. First, we determined the marble's

frequency signature by averaging all 20 measurement. Figure 2 shows what the data looks like after it has been averaged and it is a significant improvement over Figure 1. This is described in detail in Algorithm 1.

Algorithm 1: Determining the Center Frequency

Import data from `Testdata.mat`
for $j = 1 : 20$ **do**
  Extract jth measurement from `Undata`
  Compute the Fourier transform of the extracted data
  Add it to the total sum
end for
Divide sum by 20 to calculate the average
Determine the maximum frequency of the absolute value of the average
Store the maximum value and the index of the maximum value

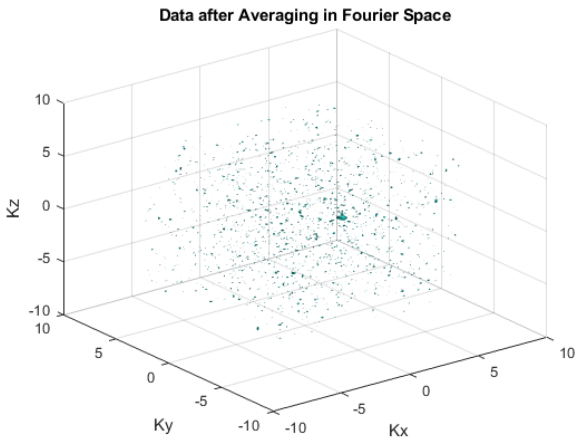

Figure 2: Isosurface of all measurements after averaging

Next, using the center frequency, we constructed an appropriate filter to de-noise the data and to determine the marbles position at each measurement. Figure 3 is an isosurface of the data after it has been filtered but before the max signal has been extracted. Even without extracting the max signal corresponding to the marble, we can observe that the filter has already constrained the location by a significant amount. This is described in detail in Algorithm 2. The filter from Equation 3 is used in this case.

Algorithm 2: Applying the filter

Import data from `Testdata.mat`
Create filter centered at the index calculated in Algorithm 1
for $j = 1 : 20$ **do**
  Extract jth measurement from `Undata`
  Compute the Fourier transform of the extracted data
  Apply filter to transformed data
  Apply the inverse transform to the filtered data
  Determine the maximum signal of the absolute value of the filtered data
  Store the position of the marble
end for

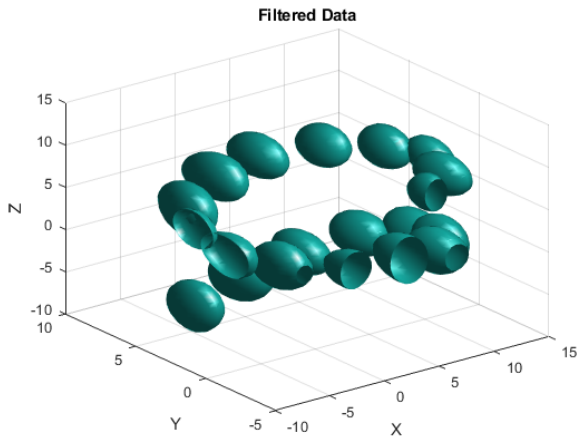

Figure 3: Isosurface of all 20 measurements after filtering

After implementing both algorithms, we are left with a matrix storing the position in (x, y, z) of the marble at each of the 20 measurements. The last row of the position matrix corresponds to the final position of the marble and is where the intense acoustic wave will be aimed at.

4 Computational Results

Table 1 shows the position of the marble at each time step after the filter has been applied.

Measurement	Position in (x,y,z)
1	(4.69, -4.22, 10.3)
2	(8.44, -2.81, 9.84)
3	(10.3, -0.469, 9.38)
4	(8.91, 1.88, 9.38)
5	(6.09, 4.22, 8.91)
6	(1.41, 5.16, 8.44)
7	(-3.28, 4.69, 7.97)
8	(-7.50, 3.28, 7.03,)
9	(-9.84, 0.938, 7.03)
10	(-9.38, -1.41, 5.63)
11	(-7.50, -3.28, 5.16)
12	(-2.81, -4.69, 3.75)
13	(2.34, -4.69, 3.28)
14	(6.56, -3.75, 1.88)
15	(9.38, -1.88, 0.938)
16	(9.84, 0.938, 0.00)
17	(7.97, 2.81, -1.41)
18	(4.22, 4.22, -3.28)
19	(-0.938, 4.69, -4.69)
20	(-5.63, 4.22, -6.09)

Table 1: The position of the marble at each measurement (rounded to 3 significant figures)

We can plot these positions to see how the marble has moved through time in Fluffy's stomach. Interestingly enough, we can observe a helical pattern (See Figure 4).The path of the marble is also similar to

the filtered data in Figure 4 which we would expect.

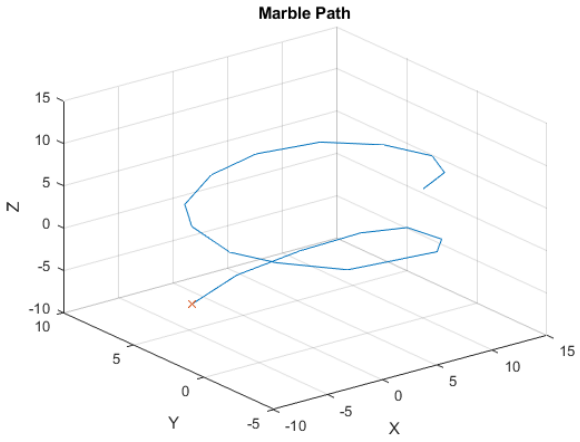

Figure 4: Marble path

From the table and the plot, we can see that we need to aim the acoustic sound wave at $(-5.625, 4.21875, -6.09375)$ in order to break up the marble and save Fluffy.

5 Summary and Conclusions

The use of Fourier analysis turned out to be highly successful in locating the marble and saving Fluffy. We were able to design an appropriate filter by averaging the noise out and used the filter to accurately determine the trajectory of the marble which was at $(-5.625, 4.21875, -6.09375)$.

References

[1] Jose Nathan Kutz. *Data-driven modeling & scientific computation: methods for complex systems & big data*. Oxford University Press, 2013.

Appendix A MATLAB Functions

Below are the key MATLAB functions implemented.

- `y = linspace(x1,x2,n)` returns a row vector of `n` evenly spaced points between `x1` and `x2`.
- `[X,Y, Z] = meshgrid(x,y,z)` returns 3-D grid coordinates based on the coordinates contained in the vectors `x`, `y` and `z`. The grid represented by the coordinates `X`, `Y` and `Z` has size `length(x)`$\times$`length(y)`$\times$`length(z)`.
- `Un(:,:,:)=reshape(Undata(j,:),n,n,n)` takes the first row of the ultrasound data and converts it to a $n \times n \times n$ array corresponding to (x, y, z).
- `[x,y,z] = ind2sub[index]` takes a linear index `index` and converts it to the equivalent subscript values corresponding to the index of an array.
- `fftn(Un)` returns the multidimentional Fourier transform of an n-dimensional array by using the fast Fourier Transform algorithm.
- `ifftn(ut` returns the multidimentional discrete inverse Fourier transform of an n-dimensional array by using the fast Fourier Transform algorithm.

- `fftshift(ut)` rearranges the Fourier transform so that the zero-frequency component is at the center. Used for plotting transformed data.
- `[max, m] = max(abs(un(:)))` returns the linear index and max value of the multidimensional array.

Appendix B MATLAB Code

```
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;
k=(2*pi/(2*L))*[0:(n/2-1) -n/2:-1]; ks=fftshift(k); %wave numbers

%mesh grids used for plotting
[X,Y,Z]=meshgrid(x,y,z);
[Kx,Ky,Kz]=meshgrid(ks,ks,ks);

%%
%Averaging the spectrum and finding the center frequency.

%reshape each row into a 64x64x64 and take the sum of each row.
sum = zeros(64,64,64);
for j = 1:20
    Un(:,:,:)=reshape(Undata(j,:),n,n,n);
    sum = sum + fftn(Un);
end

avg = sum/20; %divide total sum by 20 to get average
[cfreq,idx] = max(abs(avg(:))); %finding the center freq
[x_c,y_c,z_c] = ind2sub([64 64 64],idx); %finding the index for max freq

%%
% creating the filter and plotting position

%Where the filter will be centered at in the Fourier domain
kx_i = Kx(x_c,y_c,z_c);
ky_i = Ky(x_c,y_c,z_c);
kz_i = Kz(x_c,y_c,z_c);

%Create a gaussian filter in the fourier domain centered at
%(kx_i,ky_i,kz_i)
width = 1;
filter=exp(-width*((Kx - kx_i).^2+(Ky-ky_i).^2+(Kz-kz_i).^2));

pos = zeros(20,3); %position of the marble
for j = 1:20
    Un(:,:,:)=reshape(Undata(j,:),n,n,n);
    ut = fftn(Un);
    ut_filter = ut.*filter; %Applying the filter to each time
    un_filter = ifftn(ut_filter);
```

```
47
48      %finding where the marble is and converting it back to x,y,z
49      [max_pos, m_indx] = max(abs(un_filter(:)));
50      [x_m,y_m,z_m] = ind2sub([64 64 64], m_indx);
51      pos(j,:) = [X(x_m, y_m, z_m),Y(x_m, y_m, z_m),Z(x_m, y_m, z_m)];
52
53  end
54
55  %final position of the marble
56  final_pos = pos(20,:);
57
58  %rounding the postions for report table
59  round_pos = round(pos,3,'significant');
60
61  %plotting the trajector of the marble
62  figure(1)
63  plot3(pos(:,1),pos(:,2),pos(:,3))
64  hold on
65  plot3(final_pos(1),final_pos(2),final_pos(3),'x')
66  title('Marble Path')
67  grid on
68  xlabel('X')
69  ylabel('Y')
70  zlabel('Z')
71
72  %%
73  % Supplmentary plots for report
74
75  % Plot for the filter
76  filt_shift = fftshift(filter);
77  figure(2)
78  isosurface(Kx,Ky,Kz,filt_shift,0.4)
79  title('Filter')
80  xlabel('Kx')
81  ylabel('Ky')
82  zlabel('Kz')
83  grid on
84
85  %Plot the avg in fourier space (normalized)
86  figure(3)
87  isosurface(Kx,Ky,Kz,fftshift(abs(avg))./max(abs(avg(:))),0.5)
88  grid on
89  title('Data after Averaging in Fourier Space')
90  xlabel('Kx')
91  ylabel('Ky')
92  zlabel('Kz')
93
94  %%
95  % Plotting the initial noisy data
96
97  Un(:,:,:)=reshape(Undata(20,:),n,n,n);
98  figure(4)
99  isosurface(X,Y,Z,abs(Un),0.4)
100 axis([-10 15 -5 10 -10 15]), grid on, drawnow, title('Unfiltered Data')
```

```
xlabel('X')
ylabel('Y')
zlabel('Z')



%%
% Plotting the transformed data

Un(:,:,:)=reshape(Undata(20,:),n,n,n);
figure(5)
Ut = abs(fftn(Un));
isosurface(Kx,Ky,Kz,fftshift(Ut)./max(abs(Ut(:))),0.4)
grid on, drawnow, title('Transformed Data')
xlabel('Kx')
ylabel('Ky')
zlabel('Kz')


%%
% Plotting the filtered data (untransformed)

for j = 1:20
    Un(:,:,:)=reshape(Undata(j,:),n,n,n);
    ut2 = fftn(Un);
    ut_filter2 = ut2.*filter; %Applying the filter to each time
    un_filter2 = ifftn(ut_filter2);

    figure(6)
    isosurface(X,Y,Z,abs(un_filter2),0.1)
    grid on, drawnow, title('Filtered Data')
    axis([-10 15 -5 10 -10 15])
    xlabel('X')
    ylabel('Y')
    zlabel('Z')
end
```