AMATH 582 Homework 1

Kiana Mittelstaedt

January 24, 2020

Abstract

We have a noisy dataset containing the spatial variations of a marble inside of a dog's intestines. To save the dog, we need to calculate the trajectory of the marble so the vet knows where to focus an intense acoustic wave that will break up the marble. We apply the Fast Fourier Transform to decompose our noisy data into its frequency components. Next, we use an averaging process along with a Gaussian filter to denoise the data. Finally, we can transform our data back into its spatial domain to find the location of the marble in the dog's intestines.

1 Introduction and Overview

My dog Fluffy swallowed a marble and it has moved to his intenstines. Ultrasound data has been collected, showing the spatial variations in the area where the marble is located. However, since Fluffy is not stationary, the internal fluid movement has caused the data to be very noisy. To save Fluffy, I need to denoise this data, calculate the marble's trajectory, and report the final location to Fluffy's vet. Then, Fluffy's vet can break up the marble by focusing an intense acoustic wave on it.

2 Theoretical Background

As we learned in class, Fourier introduced the idea of representing a given function $f(x)$ by a trigonometric series of sines and cosines:

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

I will use this concept to aid in locating the marble in Fluffy's intestines by using the Fourier transform. This spectral transform will transform my spatial domain (from the ultrasound data) into a frequency domain via decomposition into sines and cosines. The Fourier transform on a function $f(x)$, as we learned from class, is defined as

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

where k is the frequency (or wave number). This integral gives the representation of $f(x)$ in all frequencies. The inverse Fourier Transform, which integrates over the frequencies to get back to the spatial domain, is defined as

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

Once the data is converted into the frequency domain, we will use an averaging technique to denoise our data. Our data has *white noise*, meaning that the noise affects all frequencies the same. Over multiple signal measurements, the *white noise* will average to zero because it can be modeled by adding a normally-distributed random variable to each spectral component. As such, we want to average over our frequency realizations to denoise and then produce a localized frequency signature.

With the localized signature in the frequency domain, we are able to locate where the signal was detected. Next, we will create a normally distributed Gaussian filter, denoted $\mathcal{F}(k)$, defined as

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

where τ denotes the bandwidth and k_0 corresponds to the frequency signature.

We will take our data, Fourier transform it, and multiply it by the Gaussian filter. If there is a signal near the frequency signature, the Gaussian filter will isolate the signal and remove any undesired frequencies that were picked up in the ultrasound detection process. With our signal extracted in the frequency domain, we will then inverse Fourier transform back to the spatial domain so we can collect information about the location of the marble and save Fluffy.

3 Algorithm Implementation and Development

First, I need to define my spatial and frequency domains so that I can use the FFT to analyze the ultrasound data that contains 20 different measurements taken in time.

1. Define the computational spatial domain L and the number of Fourier (frequency) modes, n.
2. Discretize in the spatial domain. I consider the first n points on the interval $[-L, L]$ because the FFT believes the function is 2π-periodic on this interval.
3. Define the wavenumbers k. I need to rescale the wavenumbers by $\frac{2\pi}{L}$ because the FFT assumes 2π-periodic signals.
4. Create spatial (measurement) and frequency dimensions.

Next, I need to find the frequency signature (central frequency).

1. Reshape our data into a $64 \times 64 \times 64$ array.
2. Apply the FFT to the data. This will decompose the signal into its frequency components but gives no information regarding when in time the signal occurred. This is useful because even though the signal emitted by the marble is moving around in the spatial domain, it is fixed in frequency.
3. Sum over all 20 realizations and average. This will remove the *white noise* from our data.
4. Shift the data back into the mathematically-correct domain and average. This will eliminate the *white noise*.
5. Plot the denoised data in the frequency domain.
6. Extract the frequency coordinates corresponding to the maximum signal. This is the signature (central) frequency.

To further denoise the data, I need to filter it around the central frequency. This will allow me to calculate the trajectory of the marble.

1. Build a Gaussian filter (zero mean, unit variance) around the central frequency.
2. FFT the data, shift into the mathematically-relevant domain, and multiply by the Gaussian filter. We know there exists a signal near the central frequency (a result of the averaging process), so the filter will isolate the signal input around this frequency.
3. Unshift the data and inverse FFT back to the spatial domain.
4. Locate the maximum signal detection in the spatial domain.
5. Retrieve the trajectory of the marble, with (x, y, z) coordinates for each of the 20 measurements.

The final position of the marble is the 20^{th} measurement in the trajectory. This is the location that the vet needs to focus an intense acoustic wave to break up the marble.

Algorithm 1: Applying the Gaussian Filter to Denoise

Import data from `Testdata.mat`
for $j = 1 : 20$ **do**
  Extract measurement j from `Undata`
  Reshape `Undata` to a 64-by-64-by-64 array
  FFT the array to move to frequency domain
  Shift
  Multiply by the Gaussian Filter
  Unshift
  Inverse FFT the array to return back to spatial domain
  Extract the location of the signal detection in the spatial domain after applying the filter.
  Record coordinate value.
end for

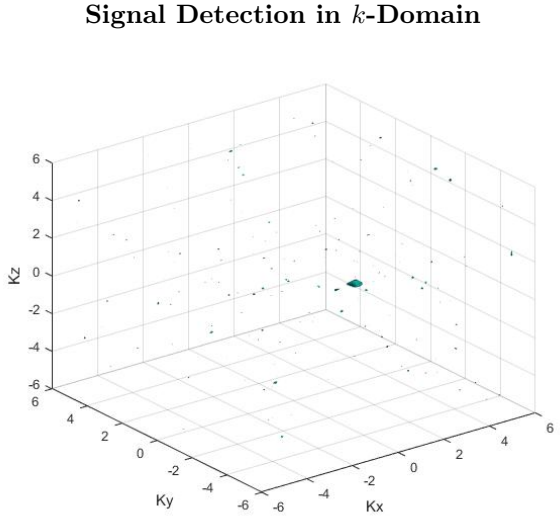

Figure 1: Signal in the frequency domain after averaging out the noise.

4 Computational Results

Problem 1:
I first needed to determine the frequency signature generated by the marble. Figure 1 shows the incoming signal from the marble after averaging in the frequency domain. The larger point in Figure 1 is the frequency signature. I took the maximum value and found that the frequency is centered at $(1.8850, -1.0472, 0)$ in the frequency domain.

Problem 2:
Next, I was tasked with computing the trajectory of the marble. To accomplish this, I created the following Gaussian filter, centered at the central frequency:

$$\mathcal{F}(k) = e^{(-0.5((Kx-1.8850)^2)+(Ky+1.0472)^2+(Kz-0)^2)}, \tag{5}$$

where Kx, Ky, Kz are coordinates in the shifted frequency domain. Figure 2 shows the 3-dimensional Gaussian filter applied to the frequency domain. I plotted the filter to ensure that it was applied to the location of the central frequency. After taking the data, transforming it, applying the filter, and transforming it back into space, I was able to detect the signal emitted by the marble in the spatial domain. This is shown in Figure 3.

To calculate the actual trajectory of the marble, I computed the maximum spatial values of the filtered data as represented in Figure 3. Doing so produced the trajectory seen in Figure 4.

Spectral Filtering

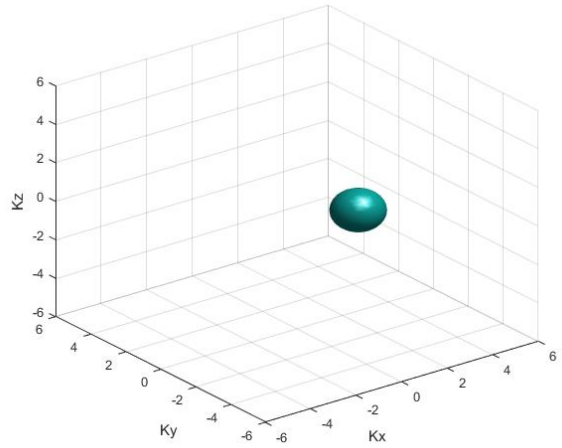

Figure 2: 3-D Gaussian filter in the frequency domain.

Spatial Detection

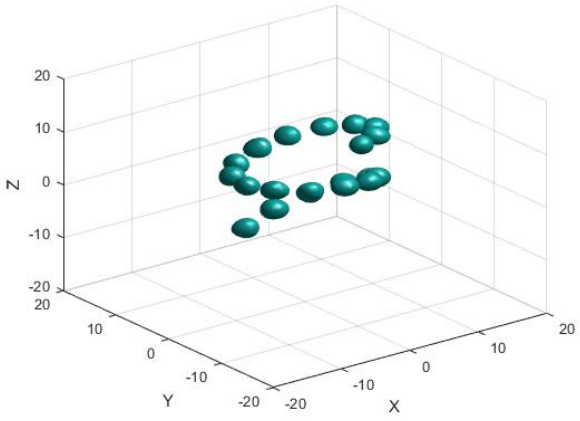

Figure 3: The suggested trajectory of the marble in the spatial domain after applying the Gaussian filter and inverse FFT.

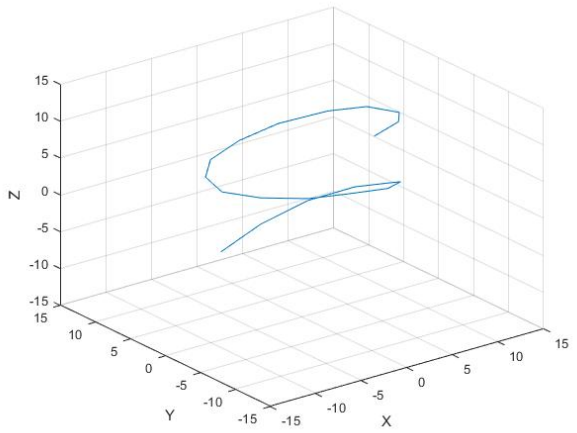

Figure 4: The actual trajectory of the marble in the spatial domain.

Problem 3:
To find the exact location of the marble, I took the position in the trajectory at the 20^{th} measurement. Thus, the vet should focus the acoustic wave at the location $(-5.6250, 4.2188, -6.0938)$ in Fluffy's intestines.

5 Summary and Conclusions

To save Fluffy, I used the Fast Fourier Transform to transform the ultrasound data. Once in the frequency domain, I was able to analyze and manipulate the data in a way that allowed me to locate the marble. I did so via first averaging over the frequency data to remove the *white noise*. Next, I applied the ideas of spectral filtering to improve the signal detection by denoising further. Using repeated measurements (20 in this case) helped me avoid a detection error. Once I transformed back into the spatial domain, I found the trajectory of the marble in Fluffy's intestines and located its final position at $(-5.6250, 4.2188, -6.0938)$. I will communicate these findings to Fluffy's vet so he can be saved!

This report can also be found on my Github Repository, linked:
https : *//github.com/kiana* − *mittelstaedt/AMATH*582

Appendix A MATLAB Functions

Here is a list of the important MATLAB functions I used with a brief implementation explanation of each.

- `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 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`).
- `reshape(X,M,N)` returns the M-by-N matrix whose elements are taken columnwise from X.
- `D = size(X)` returns a 1-by-N vector of dimension lengths for an N-dimensional array.
- `zeros(A,B,C)` is an A-by-B-by-C array of zeros.
- `fftn(X)` returns the N-dimensional discrete Fourier transform of the N-dimensional array X.
- `ifftn(X)` returns the N-dimensional inverse discrete Fourier transform of the N-dimensional array X.

- `fftshift(X)` is the FFT shift. This function swaps "half-spaces" of X along each dimension for N-dimensional arrays.
- `ifftshift(X)` is the inverse FFT shift. This function swaps "half-spaces" of X along each dimension for N-dimensional arrays.
- `[I,J] = ind2sub(SIZ,IND)` returns the arrays I and J containing the equivalent row and column subscripts corresponding to the index matrix IND for a matrix of size SIZ.
- `FV = isosurface(X,Y,Z,V,ISOVALUE)` computes isosurface geometry for data V at isosurface value ISOVALUE. Arrays (X,Y,Z) specify the points at which the data V is given.

Appendix B MATLAB Code

Below is the code I wrote to locate the marble and save Fluffy.

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

%% Define Domain

L=15; % define computational spatial domain
n=64; % Fourier/frequency modes 2^n
x2=linspace(-L,L,n+1); % define spatial discretization
x=x2(1:n); y=x; z=x; % consider first n points: periodic

k=(2*pi/(2*L))*[0:(n/2-1) -n/2:-1]; % wave #s of FFT, rescaled & shifted
ks=fftshift(k); % undo the FT shift, bring back to our domain

[X,Y,Z]=meshgrid(x,y,z); % measurement dimension
[Kx,Ky,Kz]=meshgrid(ks,ks,ks); % meshgrid frequency (wave #) domain

%% Average Over Data

Uave=zeros(n,n,n); % 3D matrix of zeros

for j=1:20
    Un(:,:,:)=reshape(Undata(j,:),n,n,n);
    UnT=fftn(Un); % Fourier transform
    Uave=Uave+UnT; % add multi-dim ft (fftn) of current data to avg
end

Uave=fftshift(Uave)/20;

% extract max frequency signature
[maxavg,i]=max(abs(Uave(:)));
[p1,p2,p3]=ind2sub(size(Uave),i); % indexes

ksx=Kx(p1,p2,p3); % max k values
ksy=Ky(p1,p2,p3);
ksz=Kz(p1,p2,p3);

close all
figure(1) % plot in frequency domain
isosurface(Kx,Ky,Kz,abs(Uave)/max(abs(Uave(:))),0.6)
```

```
axis([-6 6 -6 6 -6 6]), grid on;
xlabel('Kx'); ylabel('Ky'); zlabel('Kz');

%% Apply 3D Gaussian Filter

% filter centered around max k values ksx, ksy, ksz
gfilter=exp(-0.5*((Kx-ksx).^2 + (Ky-ksy).^2 + (Kz-ksz).^2));

close all
figure(2) % plot the Gaussian filter over frequency domain
isosurface(Kx,Ky,Kz,abs(gfilter),0.6)
axis([-6 6 -6 6 -6 6]), grid on;
xlabel('Kx'); ylabel('Ky'); zlabel('Kz');

xtr=[]; % initialize trajectory vectors
ytr=[];
ztr=[];

figure(3) % plot trajectory with filter
for j=1:20
    Un(:,:,:)=reshape(Undata(j,:,:),n,n,n);
    Unt=fftn(Un); % fft of Un
    Unts=fftshift(Unt); % shift ft
    Unf=gfilter.*(Unts); % apply the filter to shifted ft
    Unfs=ifftshift(Unf); % unshift
    Utn=ifftn(Unfs); % inverse ft

    isosurface(X,Y,Z,abs(Utn)/max(abs(Utn(:))),0.6)
    axis([-20 20 -20 20 -20 20]), grid on;
    xlabel('X'); ylabel('Y'); zlabel('Z');

    [maxU,i]=max(abs(Utn(:))); % extract max spatial values
    [p12,p22,p32]=ind2sub(size(Utn),i);

    xt=X(p12,p22,p32); % values of max in spatial domain
    yt=Y(p12,p22,p32);
    zt=Z(p12,p22,p32);

    xtr=[xtr xt]; % append to trajectory vectors
    ytr=[ytr yt];
    ztr=[ztr zt];

end

plot3(xtr(:),ytr(:),ztr(:)), grid on % plot trajectories
axis([-15 15 -15 15 -15 15]),
xlabel('X'); ylabel('Y'); zlabel('Z');

finalpos=[xtr(20) ytr(20) ztr(20)] % extract final marble position
```