Cleaning Up Ruff Data: Applying Spectral Filtering to an Ultrasound Problem

Timothy Lu

Abstract

We seek to apply spectral filtering to very noisy data in order to plot the trajectory of a marble within a dog's intestines. The goal is to sift through the noise to find where the marble is – such that we can destroy the marble and save the life of the canine in the process. We average the spectrum to find a center frequency, build a 3D gaussian filter from the center frequency data, and apply the filter to the noisy data in order to denoise and plot the trajectory of the marble within the dog.

I. Introduction and Overview

The problem is a life or death situation. My dog, Fluffy, has swallowed a marble which is moving around in his intestines. We have 20 measurements of spatial variations in the area of the intestines where the marble should be. However, because Fluffy is moving, the data is very noisy. In order to save my dog's life, I need to find the location of the marble.

The basic approach to finding the marble is as follows:

1) Apply the Fourier Transform on the noisy data to transfer the data to the frequency domain. Then adjust for the noise by averaging the spectrum to find a center frequency.
2) Filter the frequency data created using the center frequency and convert it back into the spatial domain to find the position of the marble at each measurement.
3) Use the found trajectory to find where the marble is at the end of it's path (at the 20^{th} measurement given our dataset).

II. Theoretical Background

The *spatial domain* is the domain of coordinates in space, whereas the *spectral domain* is the momentum domain – that is the spectral domain captures the momentum at a particular time. Due to the *Heisenberg Uncertainty Principle*, we cannot know the exact position or momentum of a quantum particle simultaneously (Kutz, 35).

The Fourier Transform is a form of spectral transform associated with a *Fourier Series* which "represents functions and their derivatives as sums of cosines and sines" (Kutz, 25). The *Fast-*

Fourier Transform (FFT) is an operation which performs a Fourier Transform in *O(N log N)*. For our purposes, the Fourier Transform is used to transform the noisy data into the frequency domain.

In order to account for noisy data, we filter the frequency data. This process could be referred to as *noise attenuation* via *frequency filtering*. Generally speaking, noisy data is *white noise* – "noise which effects all frequencies the same" (Kutz, 37). *Spectral filtering* denoises data such that we can "extract information at specific frequencies" (Kutz, 38). For our purposes, we find a *center frequency* where the frequency is the highest – which is also where the marble is likely to be.

To find the *center frequency* we must average the spectrum. This is a form of *spectral filtering* which denoises data that is "scrambled" by white noise. Since white noise affects all data the same, it can be represented as a normal distribution with a zero mean. The intuition then is, given multiple samples from a data set which has white noise, if the data is averaged, then the noise sums to zero. This is referred to as *averaging of the spectrum*. What is left over should be a center frequency, or a distinct frequency signal to filter around – a distinct signal where our target should be.

The type of filter we will generally apply is a *gaussian filter*. Where for a given range of values, the filter function will return 1 (else 0), thus isolating a section of the data. The following $\mathcal{F}(k)$ is a gaussian filter function of frequency k.

$$\mathcal{F}(k) = \exp(-\tau(k - k_0)^2) \tag{1}$$

where τ specifies the width of the filter and k_0 specifies where the filter is centered around. Applying this kind of spectral filter should isolate the necessary values we need to save Fluffy.

III. Algorithm Implementation and Development

The steps towards solving the problem are implemented in the following order:

1) We extract the data, and define the spatial domain and spectral domain we are working in.
2) We average the spectrum and retrieve the center frequency.
3) We construct a filter around the center frequency found in part 2.
4) We apply the filter to the transformed data and extract the spatial information.

First, we created a spatial domain according to that which is given for the data (-15 to 15 units of space) in the x, y, and z direction. Then we created a spectral domain where we rescaled the

wavenumbers (corresponding to the Fourier modes as dictated by the scale of our input data) by $\frac{2\pi}{L}$ as the FFT operation assumes 2π periodic signals. This code is the first section of MATLAB code listed in Appendix B.

Then, we averaged the domain by summing up the FFT of each sample of data. Because FFT transforms the data into the frequency domain, we do not need to worry about the difference in position of the marble – rather each sample should have the same or similar center frequency. By the definition of white noise, we should relatively accurately calculate a center frequency from this method of averaging. The result can be seen in Figure 2 where there is a center frequency found from the maximum of the average and empty space all around it (where noise was averaged out of the data). This code is in the section labeled "Averaging the Spectrum" in Appendix B.

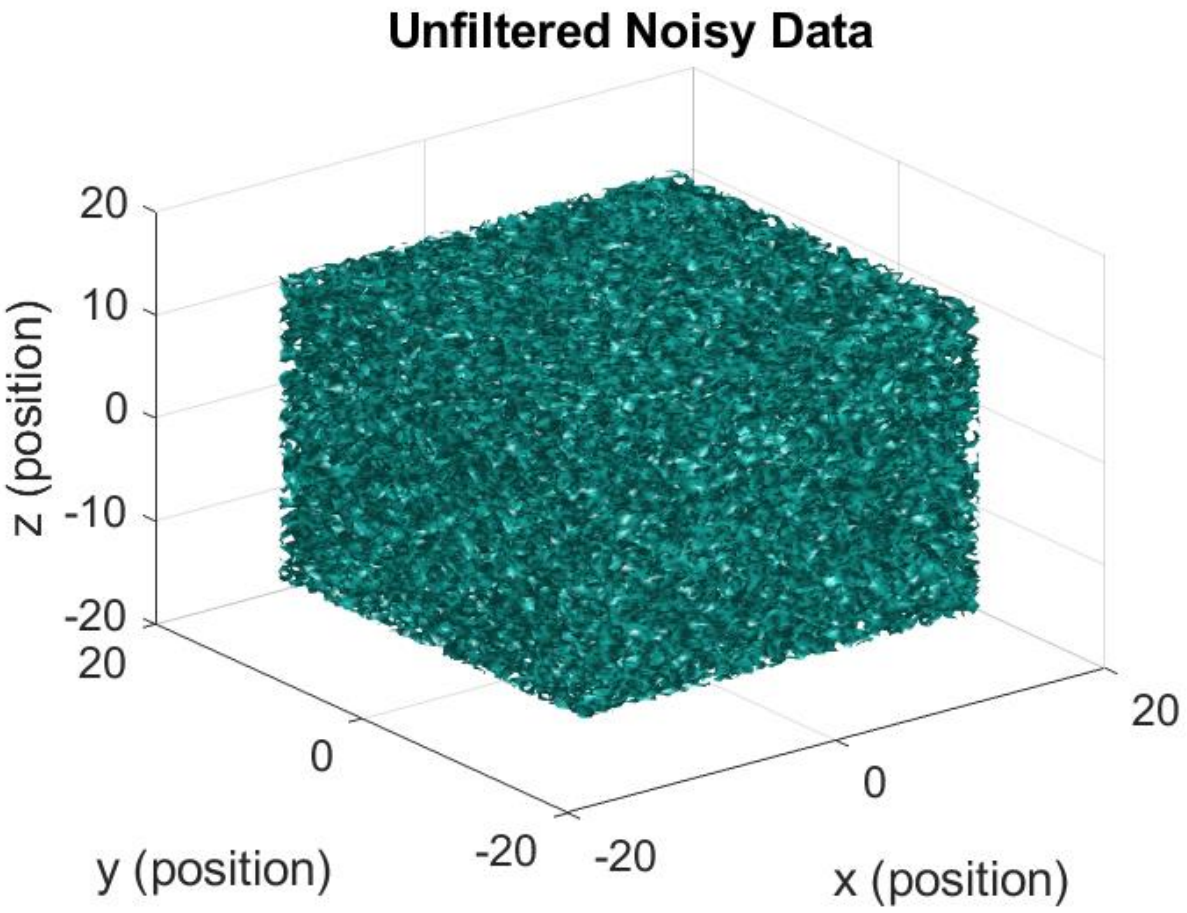

Figure 1 - A isosurface plot of the original, unfiltered and untransformed data from ultrasound

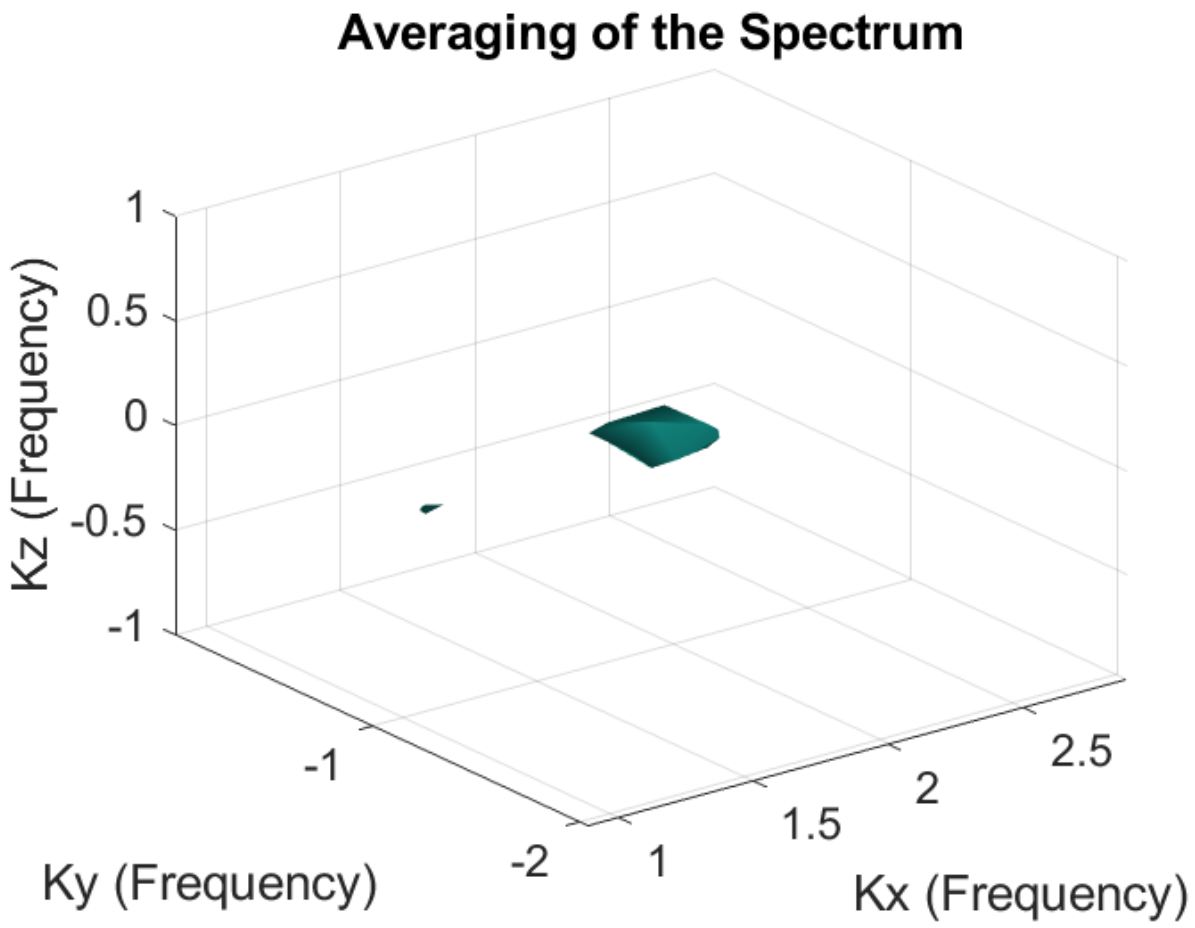

Figure 2 - The result of averaging of the spectrum (finding the center frequency)

The next step was to construct a filter around the found center frequency. We decided to use a 3-dimensional gaussian function as a filter which is the following equation (2)

$$\exp(-0.1\left[(k_x - k_{xc})^2 + \left(k_y - k_{yc}\right)^2 + (k_z - k_{zc})^2\right]) \quad (2)$$

where k_{jc} is the center frequency point of dimension $j \in \{x, y, z\}$.

The filter was then multiplied by the transform of the raw noisy data in the frequency domain and then inverse transformed (into the spatial domain) in order to find the marble location at each sample.

IV. Computational Results

From averaging of the spectrum, we found that the center frequency was approximately $(1.8850, -1.0472, 0.0000)$.

The location of the marble at the 20th moment (or 20th sample of data) was approximately (-5.6250, 4.2188, -6.0938).

V. Summary and Conclusions

The trajectory of the marble can be seen plotted in Figure 3, where each location of the marble in a unit of time is labeled (each point is labeled with the order in which it was detected). We found that, in order to save the dog, we should focus an intense acoustic wave at the position (-5.6250, 4.2188, -6.0938) in the spatial domain as this is where the marble is in the 20th data measurement.

By averaging the spectrum, we found a center frequency of the Fourier transforms of the data. Using the center frequency, we created a gaussian filter to denoise the data. Through the filter we created, the denoised data gave us clear points of data – thus allowing us to track the position of the marble through the intestines of the dog.

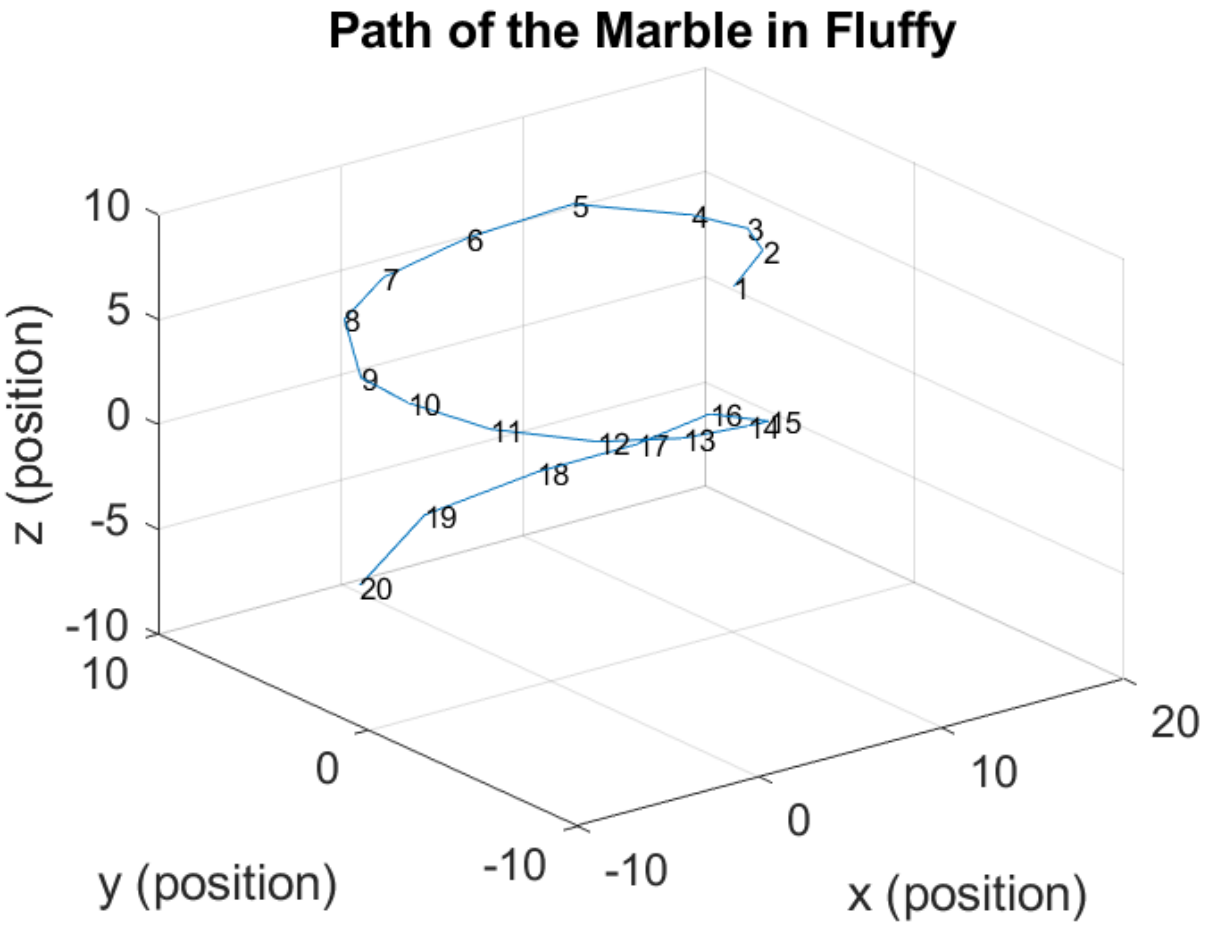

Figure 3 - The trajectory of the marble through Fluffy

Appendix A

Command	Implementation
linspace	Using the spatial domain and number of Fourier modes to create a meshgrid for displaying our data in each domain.
fftshift	Used to realign our data in the frequency domain such that all our data is correctly aligned together.
meshgrid	Used to create a coordinate system for our spatial and frequency domains.
zeros	Used to create empty vectors and matrixes to manipulate. Used to create the average matrix which summed out white-noise.
reshape	Used to shape our input data into 64 x 64 x 64 matrices such that they were in the right format.
fftn	Used to transform our data into the frequency domain. This is different from fft in that it handles multi-dimensional fft.
isosurface	Used to plot the raw noisy data, and the data in the frequency domain. Constructs a surface where data points have the same isovalue. Visualized the center frequency well.
max	Extracts the maximum from a given matrix/vector. Used to find the max frequency values (the center frequency) and their indices.
abs	Gives the absolute value of an input. Used before applying fftshift.
ind2sub	Used to get the index data from max into a usable form for access in our matrices.
ifftn	The inverse of fftn, used to transform data from the frequency domain back to the spatial domain.
plot3	Used to plot the trajectory of the marble. We found vectors of x, y, and z positions corresponding to the change in time.

Appendix B

1. Set up/starter code

```
clear all; close all; clc;
load("C:\Users\timot\Desktop\amath 482\Testdata.mat")

%% 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);
```

2. Averaging the spectrum

```
%% Averaging the Spectrum

uave = zeros(n,n,n);
for j=1:20
    Un(:,:,:)=reshape(Undata(j,:),n,n,n);
    utn=fftn(Un);
    uave=uave+utn;
end

absUave = fftshift(abs(uave));
[maximum, index] = max(absUave(:));

% Center Frequency
[I,J,K] = ind2sub([64,64,64],index)

cx = Kx(I,J,K)
cy = Ky(I,J,K)
cz = Kz(I,J,K)

close all, isosurface(Kx,Ky,Kz,fftshift(abs(uave))./max(abs(uave(:))),0.75)
axis([cx-1 cx+1 cy-1 cy+1 cz-1 cz+1]), grid on, drawnow
xlabel('Kx (Frequency)');
ylabel('Ky (Frequency)');
zlabel('Kz (Frequency)');
title('Averaging of the Spectrum');
ax = gca;
ax.FontSize = 15;
```

3. Filter the data

```
%% Filter
clc

b = 0.1;

filter = fftshift(exp(-b *((Kx-Kx(I, J, K)).^2 + (Ky-Ky(I, J, K)).^2 + (Kz-
Kz(I,J,K)).^2)));

xx = zeros(20,1);
yy = zeros(20,1);
zz = zeros(20,1);
for j=1:20
    Un(:,:,:)=reshape(Undata(j,:),n,n,n);
    uftn=fftn(Un).*filter;
    unf = ifftn(uftn);
    [maximum,index] = max(abs(unf(:)));
    [I,J,K] = ind2sub([64,64,64],index);
    xx(j) = X(I,J,K);
    yy(j) = Y(I,J,K);
    zz(j) = Z(I,J,K);
end

plot3(xx,yy,zz)
grid on
xlabel('x (position)')
ylabel('y (position)')
zlabel('z (position)')
time = [1:20]; txt = arrayfun(@num2str,time,'UniformOutput',false);
text(xx,yy,zz,txt);
title('Path of the Marble in Fluffy')
```

4. Return the location of the marble

```
%% part 3

[xx(20), yy(20), zz(20)]
```