Fourier Analysis of Ultrasound Data:

It's a Dog-Eat-Marble World

Tim Ferrin

January 24, 2020

Abstract

My dog Fluffy has swallowed a marble which has made its way into his intestines. The vet has scanned his body with an ultrasound, producing data on the marble's location at various sequential time points. This report contains an analysis of the provided data, which is noisy and inconclusive due to Fluffy's continual movements. Using fast Fourier Transform, the data is transferred into frequency space and averaged. This average is then used in a filter which is applied to the noisy data in order to determine the trajectory of the marble. Then Fluffy can be saved.

I Introduction and Overview

Fluffy has swallowed a marble which had made its way into his intestines. In order to find the marble, the vet has scanned Fluffy's body with an ultrasound at 20 different time points. This data is given to us. Due to Fluffy's movements and the movement of the internal fluid of his intestines, this data is very noisy and difficult to interpret (Figure 1). To save Fluffy, the marble must be tracked and its location at the last time step must be determined. Once its final location is determined, an intense acoustic wave is focused at the coordinates to break up the marble. Then Fluffy will be saved.

A concept of data analysis called Fourier Analysis, discussed in the "Theoretical Background Section", is used to transform the spatial data produced by the vet's ultrasound at each time step into frequency data. This transformed data is then averaged in order to find the frequency signature (center frequency) generated by the marble. Once the frequency signature is obtained, a filter is constructed and applied to the noisy, transformed data. The filtered frequency data is then inversely transformed back into the spatial domain, where interpretable information can be extracted. This process is detailed in the "Algorithm Implementation and Development" section of the report.

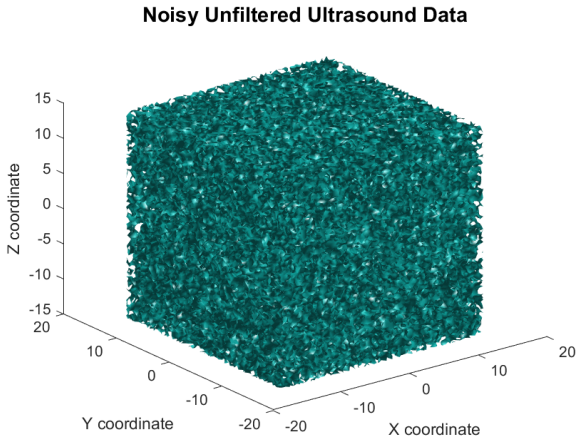

Figure 1: The first time step data plotted with an isovalue of 0.4, the same as what is used in Figure 2. There is too much noise for this data to currently produce any useful results.

II Theoretical Background

To analyze the data we must employ several elements of Fourier Analysis, the first of which is the Fourier Series. This consists of breaking down a signal function into sines and cosines of different frequencies. Let $f(x)$ be your signal function where $x \in [-\pi, \pi]$. We then rewrite $f(x)$ in the following way, where a_0, a_k, and b_k are Fourier coefficients:

$$f(x) = \frac{a_0}{2} + \sum_{k=1}^{\infty} [\, a_k \cos(kx) + b_k \sin(kx)] \tag{1}$$

Using properties of the orthogonality of sine and cosine functions, it can be shown that for $a_0, a_1, a_2, ...$ and $b_1, b_2, ...,$

$$a_k = \frac{1}{\pi} \int_{-\pi}^{\pi} f(x) \cos(kx) dx \tag{2}$$

$$b_k = \frac{1}{\pi} \int_{-\pi}^{\pi} f(x) \sin(kx) dx \tag{3}$$

It should be noted that $f(x)$ should be periodic on the interval $-\pi$ to π and that this process even works if $f(x)$ is discontinuous. To change the interval of interest from $[-\pi, \pi]$ to $[-L, L]$, formulas (1), (2), and (3) must respectively change to

$$f(x) = \frac{a_0}{2} + \sum_{k=1}^{\infty} [\, a_k \cos(\frac{k\pi x}{L}) + b_k \sin(\frac{k\pi x}{L})] \tag{4}$$

$$a_k = \frac{1}{L} \int_{-L}^{L} f(x) \cos(\frac{k\pi x}{L}) dx \tag{5}$$

$$b_k = \frac{1}{L} \int_{-L}^{L} f(x) \sin(\frac{k\pi x}{L}) dx \tag{6}$$

Using these equations, one can transfer a signal from the spatial domain to the frequency domain in what is called the Fourier Transform. This report uses the discrete Fourier Transform

due to the fact that we are not given a continuous signal, but instead a grid of discrete, evenly spaced points in the X, Y, and Z directions. For N evenly spaced points $\{x_0, x_1, ..., x_{N-1}\}$, the one dimensional Fourier and Inverse Fourier Transforms are given by

$$\hat{x}_k = \frac{1}{N} \sum_{n=0}^{N-1} x_n e^{\frac{-2\pi i k n}{N}} \tag{7}$$

$$x_k = \frac{1}{N} \sum_{k=0}^{N-1} \hat{x}_n e^{\frac{2\pi i k n}{N}} \tag{8}$$

It should finally be noted that the above formulas contain imaginary components. Although these imaginary components are useful in certain contexts, we will only consider the real components of the transforms and inverse transforms, which is why the absolute value is often taken of the signal in the code in Appendix B.

III Algorithm Implementation and Development

As above, the respective k values are the various frequencies of the sines and cosines modeling $f(x)$. Looking at evenly spaced points on a sine graph, one can see that if such a collection of points is modeled by a sine of $k+N$ frequency, it can also be modeled by a sine of k frequency. This is called "aliasing" and means that if you evaluate the discrete Fourier Transform for $k = 0, 1, 2, ..., \frac{N}{2} - 1, \frac{N}{2}, \frac{N}{2} + 1, ..., N-2, N-1$, subtracting N from the second half of the k's corresponds to the transformed values of $\hat{x}_0, \hat{x}_1, \hat{x}_2, ..., \hat{x}_{\frac{N}{2}-1}, \hat{x}_{\frac{-N}{2}}, \hat{x}_{\frac{-N}{2}+1}, ..., \hat{x}_{-2}, \hat{x}_{-1}$. This is implemented when MATLAB performs the calculations of the Fourier Transform.

All code for this report is written in MATLAB and utilizes the built in MATLAB functions fftn() and ifftn() for the heavy lifting of computing the n-dimensional Fourier and Inverse Fourier Transform respectively, as described in Appendix A. It is given by the parameters of the measurements that the signal is measured with an $L = 15$ on either side of the center, with 64 points of measurement between the boundaries of each spatial dimension. Matlab requires there be a symmetric boundary and that the number of points sampled be a power of 2. Due to the periodicity of the signal, the last point in the x vector is dropped, so to keep $n = 64$ points, $n + 1$ linearly spaced points are determined from $[-L, L]$. The k vector is then established using the stretching and aliasing as described above. The final setup includes formulating meshgrid arrays for each possible spatial and frequency combinations in the X, Y, and Z directions.

Throughout the various lines of code in Appendix B, certain variables appear to not be words but rather a random collection of letters. The table below corresponds to what each letter signifies:

Letter	**Meaning**
u	anything dealing with the ultrasound signal
n	noisy
t	transformed
f	filtered

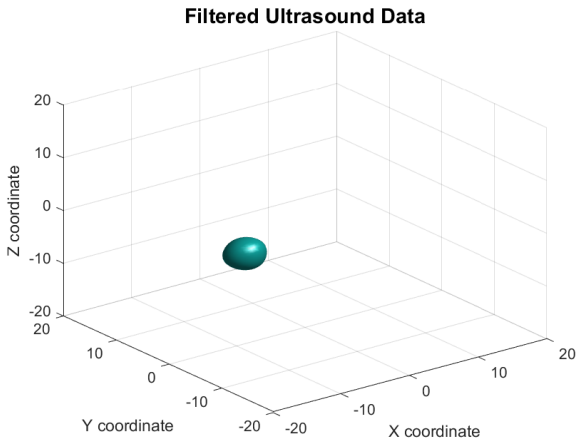

Figure 2: The last time step data plotted with an isovalue of 0.4, the same as what is used in Figure 1. Here we can clearly see the position of the marble, as the noise has been filtered out

Even though there is noise accompanying the data, due to it being random noise, random positive and negative frequencies of noise should average to 0. Thus we can average the transformed frequency domain data of each measurement, find the max absolute value frequency for each dimension, and use them as the central frequencies for our filter.

The filter that is applied to the transformed data is a simple Gaussian with a width parameter τ and a shift of k_0. A τ of 1 is chosen for this occasion. Since there is a central/signature frequency in each dimension, three filters must be applied. To apply each filter, it is simply multiplied by the transformed data. Because of the shape of the Gaussian, frequencies in the data near the central frequency are emphasized and frequencies far away from the central frequency are diminished. The three filters are combined into one filter using the properties of exponents, with

$$e^{-\tau(k-k_{x0})^2} \cdot e^{-\tau(k-k_{y0})^2} \cdot e^{-\tau(k-k_{z0})^2} = e^{-\tau((k-k_{x0})^2+(k-k_{y0})^2+(k-k_{z0})^2)} \tag{9}$$

Finally, the data filter is applied to each measurement and the filtered data is inverse Fourier transformed back into the spatial domain. The location of the center of the marble at each time step is obtained by finding where the absolute valued filtered spacial data strength is the highest. This location is then stored. Additionally, by uncommenting the last four lines of the second for loop in Appendix B, the isosurface plot can be used to plot the spatial domain normalized filtered data. This will reveal the marble's surface relative to the coordinate system at each time step (the last of which is shown in Figure 2. Leave the "close all" line commented to produce an isosurface plot of all the time steps together, showing the marble's path.

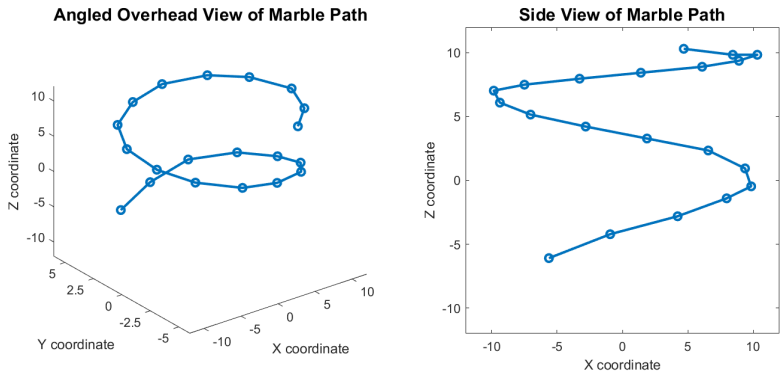

Figure 3: The path of the marble indicated by the blue dots, noting that the blue dots do not represent the size of the marble. The left subplot is of an angled overhead view and the right subplot is directly to the side, showing no y-depth. The first time location is the highest dot and the last time location is the lowest dot.

IV Computational Results

Printed out by the code in Appendix B, the three central frequencies in X, Y, and Z directions are respectively 1.885, -1.047, and 0. These correspond to the signature frequencies of the marble, or rather, what frequencies of the ultrasound signal are most strongly present in the Fourier transformed data. As mentioned above, the isosurface plot can be used to see where the marble is by computing where the transformed, filtered, inverse transformed data correspond to a specified isovalue (In this context, it was predetermined to be 0.4). Graphing the (X,Y,Z) coordinates of the marble at each time step results in the plots in Figure 3, showing the path of the marble from two different angles. The marble seems to have followed a spiral path downward, with the first measurement corresponding to the highest point and the last measurement corresponding to the lowest point. Also printed out by the code in Appendix B is the final position of the marble, (-5.625, 4.219, -6.094). This is where the intense acoustic wave should be focused in order to break up the marble and save Fluffy.

V Summary and Conclusions

The data that was given by the ultrasound was initially very noisy and no discernible information could be extracted from simply plotting it (again see Figure 1). Through Fourier analysis of the data, we averaged out the noise to create a filter for the transformed data in frequency space, applied the filter, inverse transformed the data back into spatial information, and obtained useful results. We were able to determine with a high degree of certainty where the marble was at each time step and thus where we should focus an acoustic wave at the marble to save Fluffy's life.

Even though this report focuses on a silly situation, the concepts of averaging and filtering data is incredibly useful with real world data sets. No data set is perfect, and often comes with highly distorted noise. Being able to pick through signal data and produce ac-

tionable results is something that is used across a wide variety of physical and economic situations.

Appendix A: MATLAB Functions Used and Brief Implementation Explanation

meshgrid(x,y,z) − Produces 3 multidimensional arrays to store all the possible combinations of the x, y, and z vectors.

reshape(data,n,n,n) − Used in the for loops in Appendix B to reshape the given 1 by 262,144 rows into 64 by 64 by 64 arrays.

fftn(data) − Performs the n-dimensional fast Fourier Transform, depending on the size of the signal. Hinted in the name, it is much faster than the regular discrete Fourier Transform ($\mathcal{O}(N \log N)$ vs $\mathcal{O}(N^2)$).

ifftn(data) − Performs the n-dimensional inverse fast Fourier Transform, depending on the size of the signal. As with fftn, it is faster than the discrete inverse Fourier Transform ($\mathcal{O}(N \log N)$ vs $\mathcal{O}(N^2)$).

isosurface(X,Y,Z,f,value) − Computes and plots the isosurface data from the volume data f for the given value. It is the 3-D equivalent of a 2-D contour plot.

plot3(X,Y,Z) − Plots the X, Y, and Z coordinates and connects the neighboring points with a straight line.

fftshift(data) and ifftshift(data)− shifts transformed and inversely transformed data. Used only for intermediate plotting purposes and was not needed for the figures produced in this report.

Appendix B: MATLAB Codes

```
% Setup
clear; 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];

[X,Y,Z]=meshgrid(x,y,z);
[Kx,Ky,Kz]=meshgrid(k,k,k);

```

```
% Averaging the transformed data
ave = zeros(n,n,n);
for j=1:20
    un(:,:,:)=reshape(Undata(j,:),n,n,n);
    unt = fftn(un);
    ave = ave + unt;
end
ave = ave/20;
maxInd = abs(ave) == max(abs(ave(:)));
% Central Frequencies in each spatial dimention:
kx0 = Kx(maxInd)
ky0 = Ky(maxInd)
kz0 = Kz(maxInd)

tau = 1;
% Filter for the x, y, and z directions
Filter = exp(-tau*((Kx-kx0).^2+(Ky-ky0).^2+(Kz-kz0).^2));

isoValue = .4;
path = zeros(20,3);
locInd = 0;
for j=1:20
    un(:,:,:)=reshape(Undata(j,:),n,n,n);
    unt = fftn(un);
    unft = Filter.*unt; % Applying filter
    unf = ifftn(unft);
    locInd = abs(unf) == max(abs(unf(:)));
    path(j,:) = [X(locInd) Y(locInd) Z(locInd)];
    %close all,
    %isosurface(X,Y,Z,abs(unf)/max(abs(unf(:))),isoValue)
    %axis([-20 20 -20 20 -20 20]), grid on, drawnow
    %pause(1)
end

path(20,:)

%% Figure 1
figure(1)
firstNoise = reshape(abs(Undata(1,:)),n,n,n);
isosurface(X,Y,Z,firstNoise,isoValue)
xlabel('X coordinate'), ylabel('Y coordinate'), zlabel('Z
    coordinate')
title('Noisy Unfiltered Ultrasound Data','Fontsize',14)

print(gcf,'Unfiltered_Data.png','-dpng')
```

```
57
58 %% Figure 2
59 figure(2)
60 clc
61 isosurface(X,Y,Z,abs(unf)/max(abs(unf(:))),isoValue)
62 axis([-20 20 -20 20 -20 20]), grid on, drawnow
63 xlabel('X coordinate'), ylabel('Y coordinate'), zlabel('Z
      coordinate')
64 title('Filtered Ultrasound Data','Fontsize',14)
65
66 print(gcf,'Filtered_Data.png','-dpng')
67
68 %% Figure 3
69 figure(3)
70 set(gcf, 'Position', [100, 100, 1000, 400])
71 subplot(1,2,1)
72 plot3(path(:,1),path(:,2),path(:,3),'-o','Linewidth',2)
73 axis([-12 12 -6 6 -12 12])
74 yticks(-5:2.5:5)
75 xlabel('X coordinate'), ylabel('Y coordinate'), zlabel('Z
      coordinate')
76 title('Angled Overhead View of Marble Path','Fontsize',14)
77
78 subplot(1,2,2)
79 plot(path(:,1),path(:,3),'-o','Linewidth',2)
80 axis([-12 12 -12 12])
81 xticks(-10:5:10)
82 xlabel('X coordinate'), ylabel('Z coordinate')
83 title('Side View of Marble Path','Fontsize',14)
84
85 print(gcf,'Marble_Path.png','-dpng')
```