# An Ultrasound Problem

Jason Miller
AMATH 482

January 24, 2020

**Abstract**

* In this assignment we extract the path of a marble from highly noisy ultrasound data. This is an application of signal averaging and filtering. Specifically, we convert signals taken at different points in time to their frequency spectra using Fourier Transforms, then we average the spectra to find the frequency signature of the marble, then for each spectrum, we use a Gaussian filter around that frequency signature to remove noise. Then we go back to the denoised signal domains using Inverse Fourier Transforms, where we will see the path of the marble through time.

## 1 Introduction and Overview

### 1.1 Problem Statement

Your dog Fluffy has swallowed a marble. The vet suspects that the marble is now in a small area of the intestines. By taking ultrasound scans around this area at twenty different points in time, we obtain the amount of spatial variation at each point in the area at each point in time. Because Fluffy is moving, as are his internal fluids, this data is highly noisy. In order to save Fluffy, we will locate and compute the trajectory of the marble, so that we can focus an intense acoustic wave at the marble's final location to break down the marble.

### 1.2 Computational Approach

Our approach to this problem is threefold: First, since the noise from the fluid movement is essentially random with an expectation of zero spatial variation, we can average all twenty realizations in the frequency domain, leaving just the frequencies coming from the marble.

This will give us a clear view of the marble's frequency signature. Next, we will apply a Gaussian filter around this frequency signature, which will further isolate the signal generated by the marble. Finally, we will convert this denoised data back to the spatial domain, where we will be able to clearly see the spatial movements recorded by the marble, thus showing its location at each point in time.

# 2 Theoretical Background

This task relies on two central pieces of math: *Fourier Transforms* and *Gaussian filtering.*

## 2.1 The Fourier Transform

The Fourier Transform is a continuous analogue of a Fourier series, which represents a function as a countable sum of sine and cosine waves. It is defined by:

$$\hat{f}(k) = \int_{\mathbb{R}} f(x)e^{-ikx}dx$$

Note that it is complex valued. There also exists an Inverse Fourier Transform, which converts the Fourier Transform of a function back to the original function.

$$f(x) = \int_{\mathbb{R}} \hat{f}(k)e^{ikx}dk$$

(It is worth noting that there are different definitions of the Fourier Transform, often including a scaling factor of $\frac{1}{\sqrt{2\pi}}$.) There is a similarly defined Multidimensional Fourier Transform that operates on functions from $\mathbb{R}^n$ to $\mathbb{R}$, defined by:

$$\hat{f}(k_1, \dots k_n) = \int_{\mathbb{R}^n} f(x_1, \dots x_n)e^{-i(k_1x_1+\dots+k_nx_n)}dx_1 \dots dx_n$$

In particular, we will make use of the three-dimensional transform.

Practically, the Fourier Transform represents a shift from the time domain to the frequency domain. Since it is a linear operator, we can perform our noise removal techniques in whichever space is more convenient to work in. Since our real-world data is discrete, we will use the Discrete Fourier Transform to go from our discrete time domain to our discrete frequency spectrum.

In practice, we will compute this with the Fast Fourier Transform. This is an algorithm designed by Cooley and Tukey to compute the

Discrete Fourier Transform in $O(n\log(n))$ time instead of $O(n^2)$ time. It assumes that three key conditions are met: First, it assumes that there are $2^k$ data points for some integer $k$ because the algorithm involves resursively splitting the time domain in half. Next, it assumes that our signal has periodic boundary conditions, and although the signal itself is not necessarily periodic, our boundary in the time domain does meet this condition. It also assumes that we have a domain from $-2\pi$ to $2\pi$, so we scale our frequency space accordingly.

## 2.2 Gaussian Filtering

Gaussian filtering is method to extract data around a central frequency (call it $\mathbf{k}^*$) by removing frequencies sufficiently far away from the central frequency. We accomplish this by multiplying all values in the spectrum by the filter function:

$$g(\mathbf{X}) = e^{-\tau(||\mathbf{x}-\mathbf{k}^*||_2^2)}$$

This function has the desired properties because it attains values close to zero for frequencie far away from $\mathbf{k}^*$ and is close to 1 near $\mathbf{k}^*$- just how rapidly it decays is controlled by $\tau$, and a higher value of $\tau$ results in a tighter filter. Additionally, Gaussians are eigenfunctions of the Fourier Transform, so it is computationally relatively easy to compute the Inverse Fourier Transform after multiplying the spectrum by $g(\mathbf{X})$ compared to other choices of filter functions.

# 3 Algorithm Implementation and Development

## 3.1 Setup

First, we load the .mat file containing the ultrasound data and establish our domain. The spatial domain is composed of the intersection points of a grid created by taking $n = 64$ linearly spaced points from $[-15, 15]$ in the x, y and z axes. The spatial domain is unitless in this problem. The frequency domain is a similar 3-dimensional array of grid points where each axis is from $[-\frac{n}{2}, \frac{n}{2}]$, however first each axis is **fftshift**ed due to the fact that the **fft** will swap the halves of the signal, and also scaled by $\frac{2\pi}{2L}$ because the **fft** requires a spacial domain axis ranging from $[-2\pi, 2\pi]$. We also create corresponding meshgrids for the spatial and frequency domains, where the latter is "unshifted," which we will use later.

### 3.2 Time-Averaging

Averaging the spectrum happens by first summing all of the spectra. and then taking the absolute value and dividing by twenty. The first part occurs in a loop, inside which we take a $64 \times 64 \times 64$ array that represents one realization from the ultrasound in the signal domain. Then we take the **fft** of this, and also **fftshift** because wish for the spectra to be unshifted. We add this spectrum to an overall sum of the spectra, and also save it to a $64 \times 64 \times 64 \times 20$ array of all the unshifted realization spectra, which is for later. Lastly, we take the absolute value of the (complex valued) sum and divide by twenty to get the averaged frequency domain. Since all random frequencies are now diminished, the remaining strong frequencies are from the marble (which we assume to be the only non-random frequency), so we can take the argmax of the averaged spectrum using the **max** and **ind2sub** commands. Using this method, we find that the frequency signature of the marble, $\mathbf{k}^*$, is $(1.885, -1.047, 0.000)$.

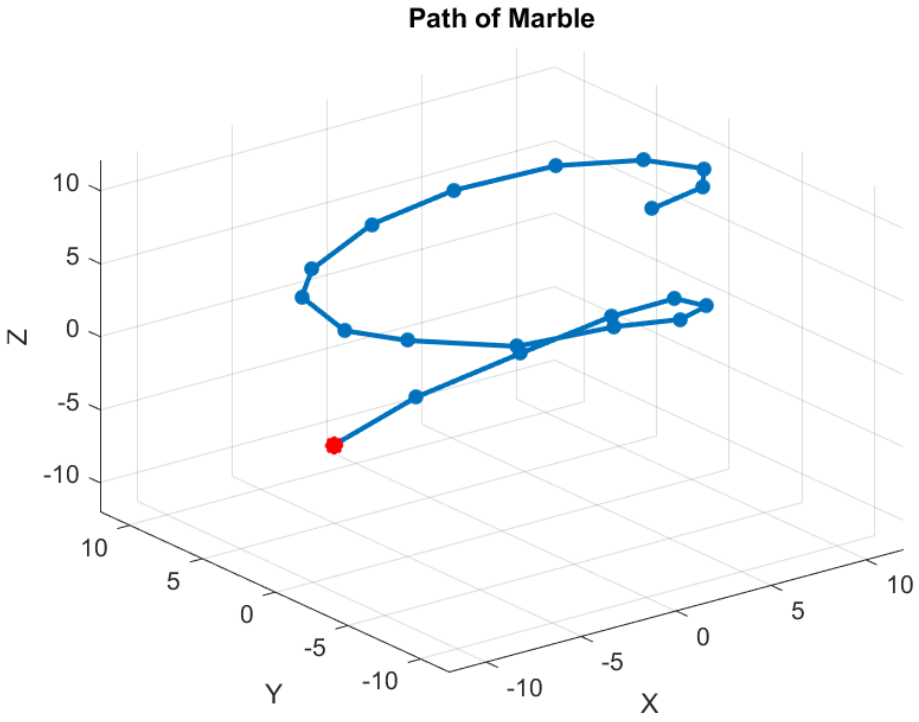
### 3.3 Filtering and Plotting

Now we apply our Gaussian filter around $\mathbf{k}^*$, making a matrix of filter values using the meshgrid for Fourier space, and multiplying each value in the array of all realization spectra by their corresponding filter values. Then for each realization, we take the **ifft** to go back to the spatial domain, and take the absolute value. This represents the denoised spacial variation data, where the strongest recorded data is coming from the marble. Therefore we take the argmax of all of these denoised realizations using the same technique as before, which gives twenty positions of the marble at each timestep. Now we plot these coordinates using **plot3**. We then play with the value of $\tau$ until the plot looks the most realistic; $\tau = .5$ seems to look nice. Lastly we make cosmetic changes to the figure, and take note of the final coordinate of the plot, which is where an intense acoustic wave should be focused.

## 4 Computational Results

Our end result is that we have the location of the marble at every point in time for which we have an ultrasound scan. These form a (relatively smooth) path that the marble traveled, as shown in Figure 1. Notably, the final data point is $(-5.625, 4.219, -6.094)$.

Figure 1: This is the path that the marble takes through Fluffy's intestines over time. The final location is shown in red.




# 5 Summary and Conclusions

Our ultimate goal was to find out where we should focus an intense acoustic wave to destroy the marble and save Fluffy. That location is simply the final coordinate of the path we found, which is $(X, Y, Z) = (-5.156, 4.219, -6.094)$.

To recap, we accomplished this task of extracting important information from a very messy dataset using Fourier Transforms, averaging, and Gaussian filtering.

# Appendix A: Function Descriptions

- abs: 2-norm
- exp: "e to the"
- fftn: n-dimensional Fast Fourier Transform
- fftshift: interchanges first and second halves of the domain

- ifftn: n-dimensional Fast Inverse Fourier Transform
- ind2sub: convert integer index to coordinate index
- linspace: creates vector with linearly spaced points
- max: maximum
- meshgrid: creates grid
- plot3: creates 3D plot
- print: saves figure
- reshape: loads data into matrix of desired dimensions
- size: dimensions of matrix
- title: titles plot
- xlabel: labels x-axis on plot
- zeros: creates matrix of zeros

# Appendix B: Code

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

%Step 0: define our domain
L=15; % spatial domain
n=64; % Fourier modes
x2=linspace(-L,L,n+1); x=x2(1:n); y=x; z=x;
k_start=(2*pi/(2*L))*[0:(n/2-1) -n/2:-1]; ks=
    fftshift(k_start);
[X,Y,Z]=meshgrid(x,y,z);
[Kx,Ky,Kz]=meshgrid(ks,ks,ks);

%Step 1: find the frequency signature
spectrum_Avg=zeros(64,64,64);
Untarr=zeros(64,64,64,20);
for j=1:20
    Un(:,:,:)=reshape(Undata(j,:),n,n,n);
    Unt=fftshift(fftn(Un));
    Untarr(:,:,:,j)=Unt;
    spectrum_Avg=spectrum_Avg+Unt;
end
spectrum_Avg=abs(spectrum_Avg)./20;
[M,I]=max(spectrum_Avg(:));
```

```
23 [Ix,Iy,Iz]=ind2sub(size(spectrum_Avg),I);
24 k_starr_x=Kx(Ix,Iy,Iz);
25 k_starr_y=Ky(Ix,Iy,Iz);
26 k_starr_z=Kz(Ix,Iy,Iz);
27 k_star=[k_starr_x k_starr_y k_starr_z];
28
29 %Step 2: apply a Gaussian filter
30 tau=.5;
31 filter=exp(-tau*((Kx-k_star(1)).^2+(Ky-k_star(2))
       .^2+(Kx-k_star(3)).^2));
32 spec_Filtered=zeros(64,64,64,20);
33 signal_Filtered=zeros(64,64,64,20);
34 for t=1:20
35       spec_Filtered(:,:,:,t)=filter.*Untarr(:,:,:,t)
             ;
36       signal_Filtered(:,:,:,t)=abs(ifftn(
             spec_Filtered(:,:,:,t)));
37 end
38
39 %Step 3: go back to signal domain and plot path
40 marble_Path=zeros(20,3);
41 for t=1:20
42       signal_Dummy=signal_Filtered(:,:,:,t);
43       [M2,J]=max(signal_Dummy(:));
44       [Sx,Sy,Sz]=ind2sub(size(signal_Filtered(:,:,:,
             t)),J);
45       marble_Path(t,1)=X(Sx,Sy,Sz);
46       marble_Path(t,2)=Y(Sx,Sy,Sz);
47       marble_Path(t,3)=Z(Sx,Sy,Sz);
48 end
49 plot3(marble_Path(:,1),marble_Path(:,2),
       marble_Path(:,3),'-*','linewidth',2)
50 hold on
51 plot3(marble_Path(20,1),marble_Path(20,2),
       marble_Path(20,3),'r*','linewidth',5)
52 xlim([-12 12]);
53 ylim([-12 12]);
54 zlim([-12 12]);
55 xlabel('X');
56 ylabel('Y');
57 zlabel('Z');
58 title('Path of Marble');
59 grid on
```

```
60 print(gcf,'-dpng','marble_path.png');
```