Homework 1

Audrey Shingleton

January 24, 2020

Abstract

My dog Fluffy has swallowed a marble. In order to save him, I need to find the trajectory of the marble inside Fluffy's intestine. A vet takes ultrasound data of where the marble is expected to be. I will use the Fast Fourier transformation, spectral averaging, and Gaussian Filtering on the 20x20 ultrasound data, taken through time concerning the spatial variations of the intestine, to determine the trajectory of the marble and save Fluffy.

I. Introduction and Overview

My dog Fluffy has swallowed a marble. The vet suspects that the marble has moved into Fluffy's intestines and records ultrasound data concerning the spatial variations of the intestines to find the marble. However, since Fluffy keeps moving, the data is highly noisy due to internal fluid movement. In order to save Fluffy, I need to locate and compute the trajectory of the marble such that it can be broken up using an intense acoustic wave.

In order to solve this problem, I will use the Fast Fourier transformation to perform the Fourier transformation on the ultrasound data. The general concept behind the Fourier Transformation is that it decomposes a time signal function into its frequency components represented as a sum of sine and cosine functions. As the noise in the data is created by random movements from Fluffy, the corresponding frequencies will be random. Thus, I can average the spectrum which will diminish the random frequencies and allow me to determine the frequency of the marble. With this, I can filter the frequency data around the marble frequency. Finally, by converting this data back to the time domain, I will be able to determine the trajectory of the marble and save Fluffy.

II. Theoretical Background

The Fourier Transform

The Fourier transform is used to decompose a time signal function into a discrete sum of sines and cosines representing its frequency components. It integrates a given function $f(x)$ dependent on time, multiplied by Euler's formula, over an infinite domain. This produces a function $F(k)$ dependent on frequency. The Fourier transform is defined as:

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

The inverse of the Fourier transform is defined as:

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

While the Fourier transform is formally defined over $x \in [-\infty, \infty]$, our computational domain is over a finite domain $x \in [-L, L]$ [1].

The Fast Fourier Transform (FFT)

The Fast Fourier transform is an algorithm to perform the Fourier transforms with a low operation count of $\mathcal{O}(N \log N)$. This is faster than the discrete Fourier transform which is performed in $\mathcal{O}(N^2)$. The algorithm achieves this lower operation count because it discretizes the range $x \in [-L, L]$ into 2^n points [1]. Additionally, the FFT shifts the data so that $x \in [0, L] \rightarrow [-L, 0]$ and $x \in [-L, 0] \rightarrow [0, L]$, multiplies every other mode by -1, and assumes a 2π periodic domain. Because the FFT assumes a 2π periodic domain, we must scale our frequencies k with $\frac{2\pi}{2L}$ [1].

Gaussian Filtering

Spectral filtering attempts to improve signal detection by denoising data by extracting information at targeted frequencies. It can remove undesired frequencies and much of the white-noise that was picked up during the detection process. This means that this technique is useful if you know what frequency to look for in the signal. A common filter, and the one used in this problem, is the Gaussian filter. This is defined as:

$$e^{-\tau(k-k_0)^2} \tag{3}$$

where τ represents the width of the filter and k_0 is the center frequency of the desired signal [1]. This filter weakens frequencies away from the k_0 and isolates the signal input near k_0.

Spectral Averaging

For data with white-noise, averaging in the frequency domain can be a powerful technique in denoising the data. The idea is that white-noise can be modeled normally with a mean of zero [1]. This means that over many signals, on average the white-noise should add to zero. This allows us to extract a clean spectral signature.

III. Algorithm Implementation and Development

Average Spectrum

The first step in solving this problem is reducing the noise in the data in order to find the center frequency of the marble. To do so, I first apply the Fast Fourier transform for each of the 20 slices of the ultrasound data recorded through time. Because the noise in the frequency data is known to be random due to Fluffy's movement, I can average the spectrum over time which will reduce the noise and allow me to extract the frequencies that correspond to the marble. I have also normalized the data using the maximum frequency value such that the data is scaled from 0 to 1.

Determine Target Frequency

Once the frequency data has been averaged and normalized, I can determine the center frequency. The frequencies that will correspond to the marble will be the maximum frequencies. Thus I can find the indices in Fourier space of the largest value to determine the target frequency.

Filter Signal

Now that I have determined the center frequency of the marble, I can use a 3D Gaussian filter on each slice of the shifted Fourier transformed ultrasound data. This will filter out the random noise around the true signal. The filter is defined as:

$$e^{-\tau((k_x-k_{x0})^2+(k_y-k_{y0})^2+(k_z-k_{z0})^2)} \tag{4}$$

where τ, the width of the filter, is set to 0.2 in this application. The variables k_x, k_y, and k_z are the 3D coordinates of the data in the frequency domain, and k_{x0}, k_{y0}, and k_{z0} are the 3D coordinates of the target frequency.

Determine Marble Trajectory

In order to determine the marble trajectory, I used the inverse Fast Fourier transform on each slice of the filtered data in the frequency domain. This gave me the ultrasound data in the time domain concerning spatial variations of the non-random marble. I then found the maximum value of each slice, representing the location of the marble, and saved its coordinates into vectors $\mathbf{x}$, $\mathbf{y}$, and $\mathbf{z}$ to

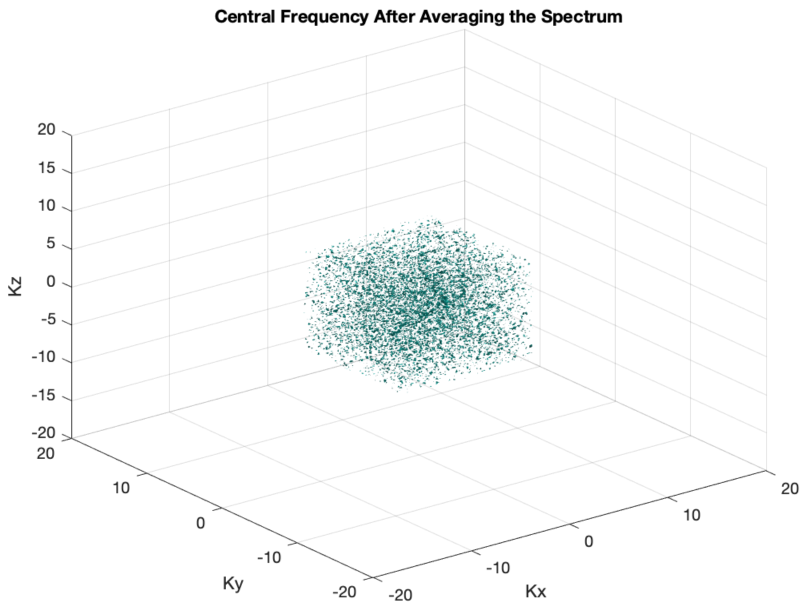

Figure 1: A visualization of the target frequency from the marble after averaging the spectrum.

plot the trajectory. The last (x, y, z) coordinate of the computed trajectory is the location of the marble.

IV. Computational Results

After averaging the spectrum over the 20 data slices, I was able to notably reduce the noise in the data (Figure 1). From this, I found that the target frequency was (1.885, -1.047, 0).

I then applied a Gaussian filter to the ultrasound data in the Fourier domain around the target frequency (Figure 2).

Once the data was filtered in the Fourier domain, I returned back to the time domain. By plotting the location of the highest intensity signal for each data slice over time, I found the trajectory of the marble (Figure 3). The final location of the marble in the intestines is shown by the green dot at (-5.625, 4.219, -6.094).

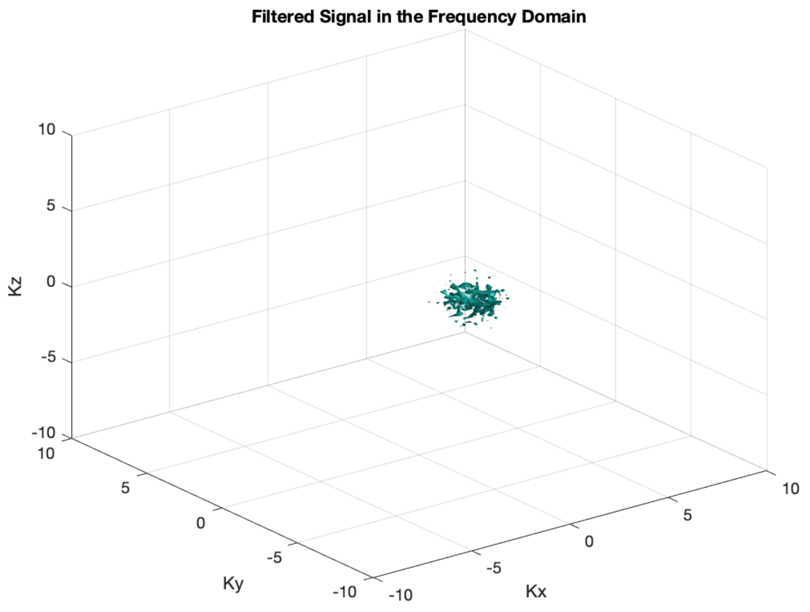

Figure 2: A visualization of the Gaussian filtered signal in the frequency domain.

V. Summary and Conclusions

The goal of this project was to determine the trajectory and location of the marble inside Fluffy the dog's intestines. To do so, 20 ultrasound measurements were taken concerning the spatial variations of the intestines. By using the Fast Fourier transform, I was able to average the spectrum to determine the center frequency of the marble, filter the noisy data around this frequency, and compute the marble trajectory. The final location of the marble in Fluffy's intestine was found to be (-5.625, 4.219, -6.094). An intense acoustic wave was focused at this point which broke up the marble and saved Fluffy.

Appendix A. MATLAB functions used and explanations

- meshgrid: Creates a 3D array based on the given coordinates as defined by the given vectors x, y, and z
- fftn: Transforms the function to the frequency domain using the Fast Fourier Transformation algorithm in n-dimensions

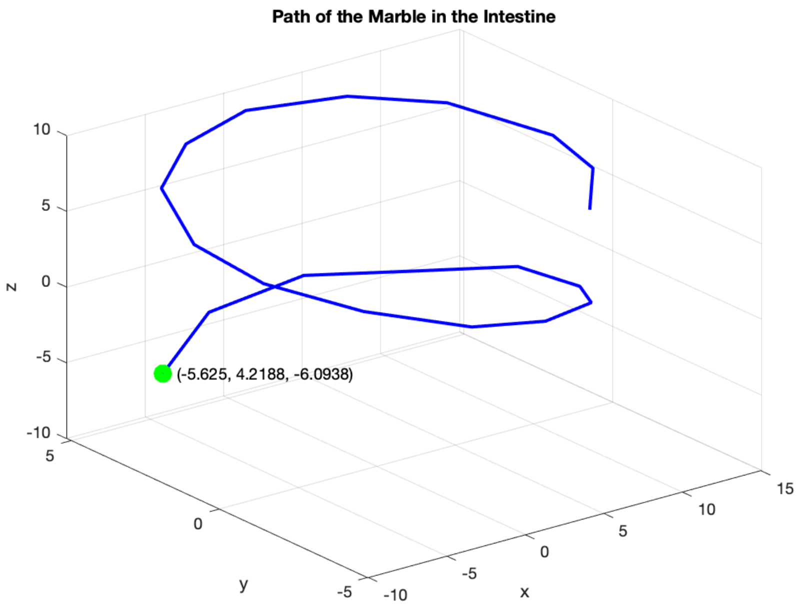

Figure 3: The marble trajectory inside Fluffy's intestine. The green dot marks the final location of the marble.

- ifftn: Transforms the function back to the time domain using the inverse Fast Fourier Transform algorithm in n-dimensions
- fftshift: Shifts the function back to its mathematically correct positions for plotting
- ifftshift: Undoes the shift from fftshift
- reshape: Reshapes the initial 2D 20x20 data into 20 3D data slices where each slice has dimension 64x64x64
- max: Returns the maximum value M in a given 3D matrix and its linear index I
- ind2sub: Returns a 3D index [x, y, z] from a linear index I

Appendix B. MATLAB codes

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

```
L=15; % Ppatial 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); % Frequency components

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

% FFT of data
Utavg = zeros(n, n, n);
N = 20;
for j=1:N
    Un(:,:,:)=reshape(Undata(j,:),n,n,n);
    % close all, isosurface(X,Y,Z,abs(Un),0.4)
    % axis([-20 20 -20 20 -20 20]), grid on, drawnow
    % pause(1)
    % Apply FFT to get signal in frequency domain
    Unt(:,:,:) = fftn(Un);
    Utavg = Utavg + Unt;
end

% Average signals
Utavg = abs(fftshift(Utavg))/N;
% Make max value 1
Utavg_norm = Utavg/max(Utavg(:));
% Frequency plot after averaging the spectrum
close all, isosurface(Kx, Ky, Kz, Utavg_norm,0.4);
title("Central Frequency After Averaging the Spectrum")
xlabel("Kx")
ylabel("Ky")
zlabel("Kz")
axis([-20 20 -20 20 -20 20]), grid on, drawnow

% Determine the frequency of interest
% Gets the index of the max (center) frequency
[M, I] = max(Utavg_norm(:)); % I is index of max
% Determines target frequency location
[I, J, K] = ind2sub(size(Utavg_norm), I);

% Center frequencies of marble
Kx0 = Kx(I, J, K);
Ky0 = Ky(I, J, K);
Kz0 = Kz(I, J, K);

% Signal filter around target frequency
```

```
filter = exp(-0.2 *((Kx - Kx0).^2 + (Ky - Ky0).^2 + (Kz - Kz0).^2));

% Filter data
x_max = zeros(1, N);
y_max = zeros(1, N);
z_max = zeros(1, N);
for j=1:N
    Un(:,:,:)=reshape(Undata(j,:),n,n,n);
    Unt(:,:,:) = fftn(Un);
    Unt_shift(:,:,:) = fftshift(Unt);
    % Apply filter
    Unt_filter = Unt_shift.*filter;
    % Go back to time domain
    Unt_filter = ifftshift(Unt_filter);
    Un_filter = ifftn(Unt_filter);
    % Get marble locations throughout time
    [M, I] = max(Un_filter(:));
    [x_m, y_m, z_m] = ind2sub(size(Un_filter), I);
    x_max(j) = X(x_m, y_m, z_m);
    y_max(j) = Y(x_m, y_m, z_m);
    z_max(j) = Z(x_m, y_m, z_m);
end

% Plot filtered signal
Unt_filter_norm = abs(Unt_filter)/max(abs(Unt_filter(:)));
close all, isosurface(Kx, Ky, Kz, fftshift(Unt_filter_norm),0.4);
title("Filtered Signal in the Frequency Domain")
xlabel("Kx")
ylabel("Ky")
zlabel("Kz")
axis([-10 10 -10 10 -10 10]), grid on, drawnow

% Plots marble trajectory
plot3(x_max, y_max, z_max, 'Color', 'b',...
    'LineWidth', 2);
grid on
% Adds labels
title('Path of the Marble in the Intestine')
xlabel('x'),
ylabel('y'),
zlabel('z')
hold on
% Gets marble location at last measurement
marble_end = [x_max(end), y_max(end), z_max(end)];
% Plots marble
plot3(marble_end(1), marble_end(2), marble_end(3), '.', ...
    'Color', 'g', 'markersize', 40)
% Adds marble location to plot
text(marble_end(1), marble_end(2), marble_end(3),...
    ['    (' num2str(marble_end(1))  ', ' num2str(marble_end(2))  ', ' ...
    num2str(marble_end(3))  ')'])
```

References

[1] Kutz, Nathan J. “Computational Methods for Data Analysis.” *In Data-Driven Modeling Scientific Computation: Methods for Complex Systems Big Data.*. Oxford University Press, 2013.