An Ultrasound Problem

Amanda Lee

Abstract

My dog has swallowed a marble. I need to locate the marble using noisy ultrasound data. I first average out the spectrum to determine the frequency generated by the marble. Next, I use a Gaussian filter the data around the center frequency of the marble to find the location of the marble at each point in time. Finally, I use this information to find the last location of the marble, so I know where to focus an intense acoustic wave to break it up.

Introduction and Overview

My dog Fluffy has swallowed a marble. The vet thinks that the marble has worked its way down into his intestines. Ultrasound data in twenty time intervals was gathered in the region of the intestines where the marble is believed to be. The data is very noisy, however, because of the movement of fluid through the intestines. I need to use certain methods to denoise the data and use it to figure out the path of the marble. Once I know the path of the marble, I will know where to focus an intense acoustic wave to break up the marble.

Theoretical Background

These are the methods I will use to help me denoise the data.

Fourier transform

The Fourier transform is a way of converting a function into a series of sines and cosines. This lets us analyze data in the frequency domain by being able to see how much of each frequency is present in a signal.

The Fourier transform is mathematically defined as the following

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

The forward and backward Fourier transforms lets us convert a signal to and from the frequency domain. An algorithm for doing this quickly using a computer is called the Fast Fourier Transform (FFT). I will be using the FFT related functions in MATLAB. FFT assumes 2π periodic signals, so the frequencies given by FFT must be rescaled by $\frac{2\pi}{L}$, where L is the duration of time.

Averaging

Averaging is a method of reducing noise by taking the average of the intensities of the frequencies over time. This works because random noise is normally distributed, meaning that the mean of noise is zero. After the average is taken over the sum of the data points, we will be able to see the signal much more clearly because most of the noise will be averaged out. It is important to note that the average must be taken in the frequency domain. To do this, the Fourier Transform of the signal must be taken first. Averaging is useful for getting the frequency of the signal with reduced noise, but is not sufficient if the signal is coming from something that is moving. Because averaging is done in the frequency domain, the information about how the signal changes over time is lost.

Gaussian Filter

A Gaussian Filter is another method of noise reduction. It works by multiplying the signal in the frequency domain by a gaussian centered around the frequency of interest. This has the effect of zooming in on the frequency of interest and getting rid of information about undesired frequencies. Once the inverse Fourier Transform of the filtered signal is taken, it becomes much easier to see the signal because much of the noise will be filtered out.

The Gaussian Filter is defined as

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

Where τ is the width of the filter, k is the wavenumber, and k_0 is the center-frequency or the frequency of interest. For three dimensional data, such as in this case, the filter looks like this

$$\mathcal{F}(k) = e^{-\tau(k_x-k_{0x})^2(k_y-k_{0y})^2(k_z-k_{0z})^2}$$

Algorithm Implementation and Development

To start off, I made half of the length of the spatial domain to 15 and set the number of Fourier modes to 64. I then make vectors of the possible x, y, and z values of every point in the spatial domain and of the wavenumbers in the frequency domain. The wavenumbers are rescaled by $\frac{2\pi}{L}$, where L is the length of the spatial domain.

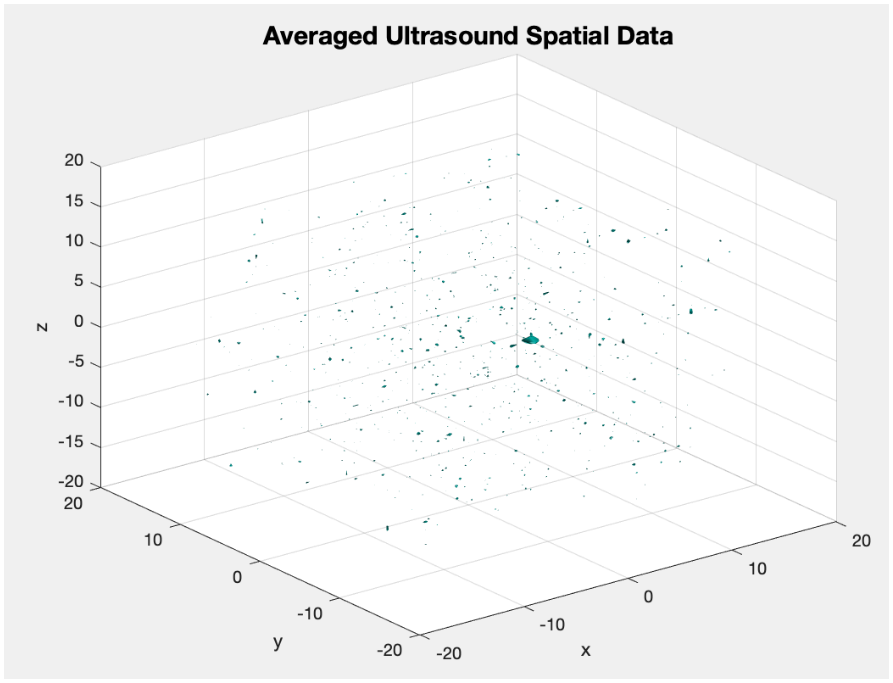

Figure 1: The isosurface of the averaged data. The marble is distinguishable among the noise.

The main goal is to know where the marble is based on how the marble is moving, but the data is too noisy to make out the marble. A gaussian filter would help us see the trajectory of the marble and reduce noise, but we need to know the frequency signature of the marble first. We can do this by averaging out the data in the frequency domain. This will reduce the noise enough to see the center frequency of the marble, but will not tell us how the marble is moving because the data in the frequency domain will not have time information. To do this, I made a matrix to hold the sum of the signal information over each point in time. Then, I looped over each time segment and reshaped the data from a vector of 262,144 numbers into a 64x64x64 matrix to make it easier to distinguish between the x, y, and z dimensions. Then, I took the FFT of the signal and added the result to the matrix. After this, I divided the absolute value of the FFT of the matrix by 20, which is the number of data points. The most prominent frequency would be the frequency of the marble, so I took the index of the maximum value and used it to get the frequencies from the vectors of the wavenumbers. The marble is now visible in the isosurface of the averaged data (Figure 1).

Since we now know the frequency of the marble, we can use a gaussian filter centered on that frequency. I chose to use a τ value of 0.2. I then defined a filter according to the equation of a 3D gaussian filter and shifted the values to that the zero values were in the center to match up with the data. To filter each data point in time, I looped over the original data again and multiplied the filter by each signal in frequency space. The goal is to locate the marble at each point in time, so took the index of the maximum of the inverse FFT of the filtered signal and then found the corresponding points in the spatial domain. The location of the marble will be the last set of coordinates.

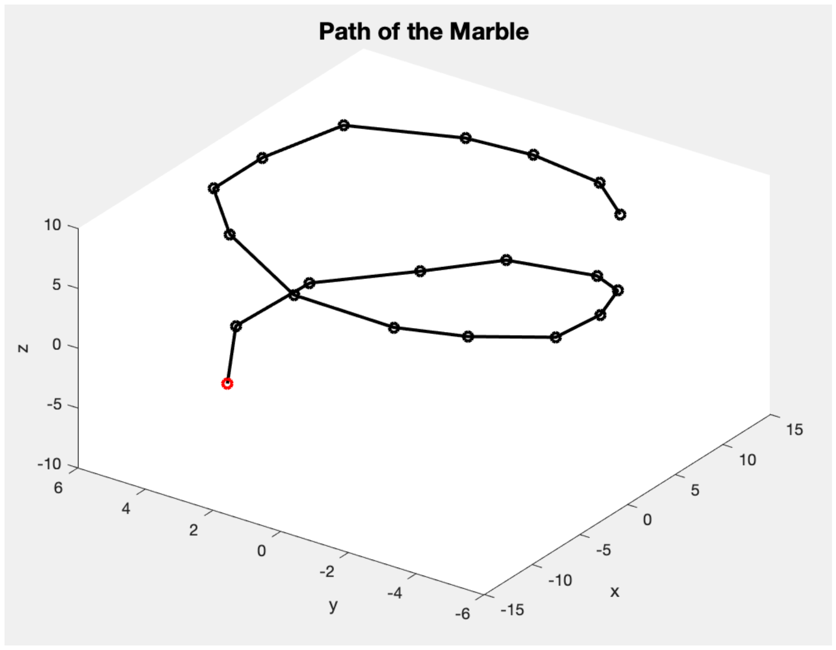

Figure 2: A Plot of the position of the marble over time. The last point is colored red.

Computational Results

The x, y, and z frequencies of the marble were determined to be x = 1.8850, y = -1.0472, z = 0
The plot of the locations of the marble over time reveals its trajectory. The marble is moving downwards in a helical motion as seen in Figure 2.

The last point of the marble is at (-5.6250, 4.2188,-6.0938).

Summary and Conclusion

To help my dog Fluffy, I had to locate the marble in his intestines using ultrasound data. Unfortunately, the data was very noisy, requiring the use noise reduction techniques. I first found the frequency of the marble by averaging the data in the frequency domain and getting the maximum of the average data. I then used the frequency of the marble as the central frequency in a gaussian filter and applied it to the data at each point in time. This let me find the location of the marble at every time point. I then plotted the path of the marble. The last location of the marble is (-5.6250, 4.2188,-6.0938). This is where an intense acoustic wave should be focused to break up the marble.

Appendix A. MATLAB Functions Used

`Y=fftn(X)` returns the multidimensional Fast Fourier transform of an array. A one dimensional Fourier transform is done along each dimension of `X`. This is used to convert the signal to the frequency domain.

`Y=fftshift(X)` shifts the zero-frequency component of `X` to the center of the array. This is used on the filter so that the values are arranged properly.

`Y=ifftn(X)` returns the inverse of the multidimensional Fast Fourier transform of an array. This is used to convert the signal from the frequency domain to the spatial domain.

`isosurface(X,Y,Z,V,isovalue)` is a way of showing three dimensional data, `V`, at points on a grid defined by the values of `X`, `Y`, and `Z`. The points at the specified isovalue are connected together. This is used to visualize the averaged data.

`[M,I]=max(A,[],'all','linear')` returns the maximum value, `M`, and linear index, `I`, of a multidimensional array. This is used to get the indices of the marble frequency and the positions of the marble from the averaged and filtered data.

`[X,Y,Z]=meshgrid(x,y,z)` returns three dimensional grid coordinates defined by the vectors `x`, `y`, and `z`. This is used to create the grid of coordinates in the spatial and frequency domains.

`B=reshape(A,sz)` reshapes `A` into an array of size `sz`. This is used to reshape the data at each time point from a 262,144 number long vector to a 64x64x64 array.

Appendix B. Matlab Code

```
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]; ks=fftshift(k);

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

% Create a matrix with the sum of all the data points over time
Uavg = zeros(n,n,n);
for j=1:20
     Un(:,:,:)=reshape(Undata(j,:),n,n,n);
     Uf = fftn(Un);
     Uavg = Uavg + Uf;
end

% Divide the sum by the number of time points and shift values
Uavg = abs(fftshift(Uavg))./ 20;

% Show the image of the averaged data
isosurface(X,Y,Z,abs(Uavg),150)
axis([-20 20 -20 20 -20 20]), grid on, drawnow
xlabel('x'), ylabel('y'), zlabel('z')
title('Averaged Ultrasound Spatial Data', 'Fontsize', 15)
%%
% Get maximum frequency
[M,I] = max(abs(Uavg),[],'all','linear');
k0 = [Kx(I),Ky(I),Kz(I)];

% Define filter
tau = 0.01;
filter = exp(-tau*((Kx-k0(1)).^2.*(Ky-k0(2)).^2.*(Kz-k0(3)).^2));
filter = fftshift(filter);

% Create vectors to hold coordinates
X_coords = zeros(20,0);
Y_coords = zeros(20,0);
Z_coords = zeros(20,0);
path = zeros(20,3);
for j=1:20
     % Reshape data
     Un(:,:,:)=reshape(Undata(j,:),n,n,n);
     % Apply the filter to signal in frequency space
     Uf = fftn(Un);
     unft = filter.*Uf;
     % Construct the filtered signal in time domain
     unf=ifftn(unft);
     [M,I] = max(abs(unf),[],'all','linear');
     % Get coordinates of maximun frequency
     X_coords(j) = X(I);
     Y_coords(j) = Y(I);
     Z_coords(j) = Z(I);
end

% Plot the trajectory of the marble
plot3(X_coords,Y_coords,Z_coords,'ko-', 'Linewidth', 2)
xlabel('x'), ylabel('y'), zlabel('z')
title('Path of the Marble', 'Fontsize', 15)

% Plot the last location of marble
hold on
plot3(X_coords(20),Y_coords(20),Z_coords(20),'ro', 'Linewidth', 2)
```