An Ultrasound Problem

Lahari Gorantla
AMATH 482

Abstract

While it is beneficial to record biological data, the result is often noisy due to the environment of the body and the limitations of the recording device. The problem this paper is addressing is finding the trajectory of a marble in a dog's intestine via ultrasound data gathered in a 3D spatial domain. A tool explored in this paper is using computational processing by implementing the Fourier transform onto the 3D data to transform it to the frequency domain, to identify the signature frequency, and to filter the noise around it to decrease the noise and find the desired data.

I. Introduction and Overview

The marble that Fluffy ate is currently moving around in his intestines. The vet utilized an ultrasound to gather 3D spatial data to locate the whereabouts of the marble. The ultrasound works by transmitting high-frequency sound waves into the body. When the wave hits a tissue boundary, waves are reflected to the probe. Depending on the distance from the ultrasound transducer to the boundary, a 2D image is visualized on the screen. A 3D image can be formed via a process known "surface rending" where data from multiple reflections are gathered and interpreted as a 3D image.

The vet has collected ultrasound data in a small area in the intestine for a time course of 20 time points with 64 x 64 x 64 points in x, y, and z dimensions. However, Fluffy kept moving and the intestinal fluid movement through the intestines generated highly noisy data. To reduce the noise and find the location of the marble, the data needs can be cleaned up by computational processing.

II. Theoretical Background

Fourier Transform

Spectral transforms, such as Fourier transforms, are of the most powerful and efficient techniques for solving a myriad of physical, biological, and engineering problems. This transform can take a 2π-periodic signal over a time course or a signal over a spatial domain and decompose it into its constituent frequencies in the frequency domain, denoted by *k*.

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

The domain for the signal function is $x \in [-\infty,\infty]$ with an eigenfunction expansion over all continuous wavenumbers *k*. Therefore, equation 1 has a limitation to periodic and infinitely continuous signals. This is not an attribute of the computational domain of collected biological data. The computation domain has a limited finite domain of x ∈ [-L,L], and a discrete sum of eigenfunctions and associated wavenumbers.

Discrete Fourier Transform

The Discrete Fourier Transform (DFT), a variant of the Fourier Transform, can be used to transform discrete, uniformly-spaced spatial or time course data into a discrete frequency representation. The N represents the total number of samples, n represents the position of the point in the discretized vector, and *k* represents the wavenumber of the frequency domain. The X[k] represents the strength of the frequency.

$$X[k] = \sum_{n=0}^{N-1} x_n e^{\frac{-2\pi ikn}{N}} \qquad (2)$$

However, the DFT is very slow, as it is an O(N^2) operation. Therefore, MATLAB utilizes the *fft* function which computes the DFT of the input using a fast Fourier transform algorithm, a O(*NlogN*) operation.

In the case of marble, 3D data is gathered. The DFT equation for that takes the summation of spatial vectors in all x, y, and z dimensions.

Inverse Discrete Fourier Transform

The inverse Fourier Transform transforms data from the frequency domain to the spatial or time domain.

$$X_n = \frac{1}{N} \sum_{k=0}^{N-1} x_k e^{\frac{2\pi ikn}{N}} \qquad (3)$$

Averaging the Spectrum

To create improved time-frequency resolution, noise can be reduced by averaging the spectrum. White noise should add up to 0, when taken a summation of over multiple signals.

Gaussian Filtering

The Gaussian filter is a commonly utilized spectral filter in signal processing. It is a low pass filter that eliminates high-frequency components in the system of interest. However, the user needs to know what the desired frequency being investigated is, for the filter to work.

$$y = \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{(x-\mu)^2}{2\sigma^2}} \qquad (4)$$

$$\mathcal{F}(k) = e^{(-\tau(k-k_o)^2)} \qquad (5)$$

Equation 4 represents the Gaussian curve with a mean of μ and a standard deviation of σ. Equation 5 is the Gaussian filter used for spectral filtering. The τ measures the bandwidth of the filter and the k_o represents the central frequency of the desired signal field. When the Gaussian filter, with the desired frequency centered at k_o, is multiplied with the data in the frequency domain, the frequencies further away from the central frequency are strongly attenuated.

III. Algorithm Implementation and Development

The software used to develop the algorithm was MATLAB_R2019a.

The ultrasound dataset consists of x, y, and z spatial points over the time course of 20 time points. The spatial points, also known as Fourier nodes, are 64. A value of 2 to the power of some *n* gives an optimized performance; since 64 is equivalent to 2^6 and therefore gives an ideal performance. The dataset is a 20 x 262144 matrix because each time row has data for all 64 x 64 x 64 nodes for the x, y, and z dimensions. The size of the spatial domain of the ultrasound probe is the interval [-15, 15], where units were not specified. There are 64 Fourier nodes so that is the increment used to build the 3D spatial domain.

Given the number of Fourier nodes, the frequency domain spans from [-32, 31]. The *k* is vectorized in MATLAB as *k* = 0, 1, 2… 31, -32, 31… -1. The Fourier transform needs to be rescaled to the 2π domain and so the vector of wavenumber *k* is multiplied by $\frac{2\pi}{2L}$, where L is 15 for this problem. Once the vector *k* is created, it needs to be shifted so that x ∈ [0, L] becomes [-L,0] and [-L,0] becomes [0, L].

To create a 3D representation of the data, the x, y, and z spatial vectors and kx, ky, and kz shifted frequency vectors are initialized into three dimensional arrays, [X, Y, Z] and [Kx, Ky, Kz], via *meshgrid*.

To locate the marble in the intestine, there are three main steps: finding the center frequency of the marble, building and applying the filter, and finding the spatial location at the last time point.

Finding the Center Frequency *(see code lines 13-30)*

To reshape the 20 x 262144 dataset, an iterative process was utilized to reshape the matrix to a 64 x 64 x 64 image for 20 time points. The average spectrum can be found by calculating the summation of the fast Fourier transformed data over the time course of 20 time points, shifting to match the indices of the shifted *k* values, and then dividing by 20.

The index of the maximum strength value in the average spectrum is the signature frequency of the marble. To bypass the problem of finding a maximum in 3D, the values in the 3D average spectrum can be reorganized into 1 column vector. Then, once the max is found, the grid can be rebuilt to identify the indices corresponding to with the max value. Then, those indices can be used to identify the indices of the original Kx, Ky, Kz values.

The corresponding frequencies of the max strength value found were: 1.8850, -1.0472, and 0 for kx, ky, and kz, respectively.

Building and Applying the filter *(see code lines 32-53)*

Utilizing the values found above for center frequency, a 3D Gaussian filter can be built.

$$\mathcal{F}(k) = e^{(-\tau[(kx-k_{xo})^2 + [(ky-k_{yo})^2 + [(kz-k_{zo})^2)]} \qquad (6)$$

The τ used was 0.2; this value was arbitrarily chosen. The filter takes the difference of the center frequencies and corresponding Kx, Ky, Kz frequency domains.

Then the shifted and fast Fourier transformed ultrasound data can be multiplied with the filter to attenuate frequencies that do not belong to the marble. Then the data is converted to the spatial domain and the spatial coordinates associated with the maximum value are identified; this step uses the same methodology as finding the center frequency. This process is done for the 20 time points to identify the trajectory of the marble.

Locating the Last Position of the Marble *(see code lines 55-68)*

To map the trajectory of marble for 20 time points, the x, y, and z position of the marble can be saved in a vector. The last value in the vector responds to the last location of the marble.

IV. Computational Results

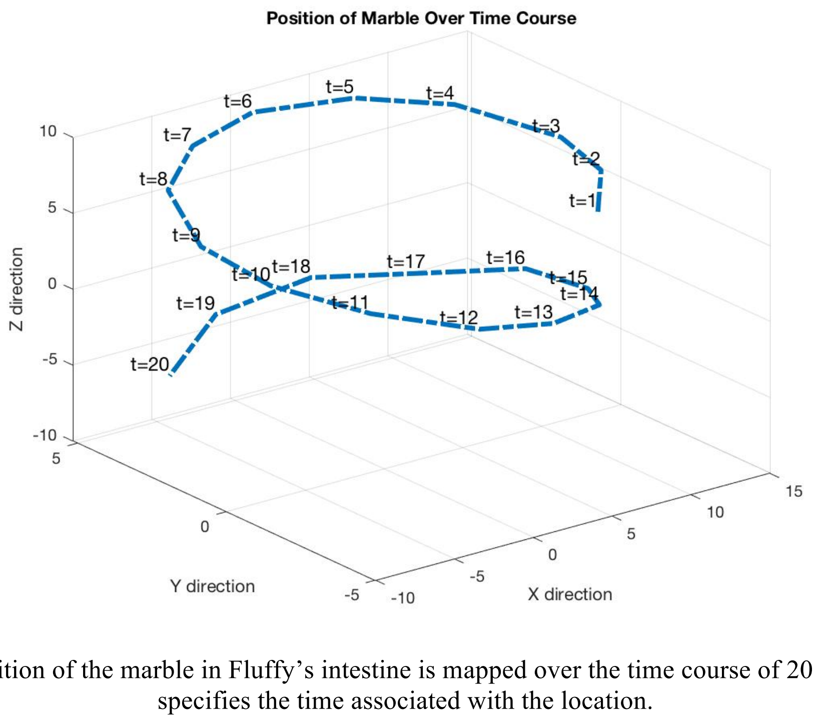

Figure 1. The position of the marble in Fluffy's intestine is mapped over the time course of 20 time points. The t specifies the time associated with the location.

As seen in Figure 1, there appears to be a spiral pattern in the trajectory of the marble. At the 20th time point, the marble is at location (-5.6250, 4.2188, -6.0938) in the intestine.

V. Summary and Conclusion

The final position of the marble is at (-5.6250, 4.2188, -6.0938) in the intestine and so this is where the acoustic wave needs to be focused to break the marble. This information was found by identifying the center frequencies (1.8850, -1.0472, 0), for application of the Gaussian filter.

Signals recorded in biological systems tend to have noise from the environment and the device recording. In the frequency domain, one can average the spectrum to eliminate some noise and identify the center frequency. By finding the center frequency, a Gaussian filter can be applied to the signal via multiplication to attenuate the noise while preserving that amplitude of the center frequency and of the surrounding frequencies within the bandwidth of the filter. Then this information can be processed in the spatial domain to identify the location of the marble over a time course. This demonstrates the power of spectral analysis.

Appendix A. MATLAB functions used and brief implementation explanation

meshgrid(x,y,z): Depending on the number of inputs into the function, a grid coordinate system of that dimension is initialized. In this case, a 3D grid coordinate system is initialized.

reshape(A,n,n):
Reshapes the size of a matrix, A, to the dimensions specified by the following inputs, denoted by n. Reshaped the ultrasound data into a 3D dataset.

fftn(x):
Performs a multidimensional fast Fourier transform on dataset x.

fftshift(x):
Shifts the zero frequency to the middle of the system and switches the left-side and right-side of the frequency plot.

ind2sub(sz,ind):
Outputs the location for the indices located in a matrix of a certain size. This was used to find the center frequency after vectorizing the 64 x 64 x 64 data into 1 column to find maximum. It reshaped into column back into a 64 x 64 x64 coordinate grid and outputted those indices.

ifftn(x):
Performs an inverse multidimensional fast Fourier transform on dataset x.

plot3:
Plots the coordinates in 3D. Used to find trajectory of the marble.

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; %spatial domain discretization
k=(2*pi/(2*L))*[0:(n/2-1) -n/2:-1]; ks=fftshift(k); %freq components of fft
[X,Y,Z]=meshgrid(x,y,z);
[Kx,Ky,Kz]=meshgrid(ks,ks,ks); %makes a coordinate system

ave_sig = zeros(n,n,n); %creates a 3D array of 0s

%Averaging the spectrum
for j=1:20
    Un(:,:,:)=reshape(Undata(j,:),n,n,n);
    ut = fftn(Un);
    ave_sig = ave_sig + ut;
end
ave_sig = abs((fftshift(ave_sig)))./20;

%finding the max value, finding corresponding indices
[max_amp,index] = max(ave_sig(:));
[center_x,center_y,center_z] = ind2sub([n,n,n],index);

%utilizing fake indices to find the correct indicis
center_kx = Kx(center_x,center_y,center_z);
center_ky = Ky(center_x,center_y,center_z);
center_kz = Kz(center_x,center_y,center_z);

% Guassian Filter
filter = exp(-0.2*((Kx - center_kx).^2 + (Ky - center_ky).^2 + (Kz - center_kz).^2));

pos_x = zeros(20,1);
pos_y = zeros(20,1);
pos_z = zeros(20,1);

%applying the filter on each time point
for j=1:20

    Un(:,:,:)=reshape(Undata(j,:),n,n,n);
    ut = fftshift(fftn(Un));
    filtered_sig = filter.*ut;
    og = (ifftn(filtered_sig));

    [max_sig,Index] = max(og(:)); %finding max position
    [center_X,center_Y,center_Z] = ind2sub(size(og),Index);
    pos_x(j) = X(center_X,center_Y,center_Z);
    pos_y(j) = Y(center_X,center_Y,center_Z);
    pos_z(j) = Z(center_X,center_Y,center_Z);

end

%mapping the trajectory
plot3(pos_x,pos_y,pos_z,'-.','Linewidth',3); grid on;
title('Position of Marble Over Time Course');
xlabel('X direction');ylabel('Y direction');zlabel('Z direction');
labels = {'t=1','t=2','t=3','t=4','t=5','t=6','t=7','t=8','t=9'...
    't=10','t=11','t=12','t=13','t=14','t=15','t=16','t=17','t=18'...
    't=19','t=20'};
text(pos_x,pos_y,pos_z,labels,'VerticalAlignment','bottom',...
    'HorizontalAlignment','right','FontSize',12)

%where accoustic wave should be focused
lastx = pos_x(end);
lasty = pos_y(end);
lastz = pos_z(end);
```