Amath482 – Winter 2020

An Ultrasound Problem

Chenxi Di (1663044)

January 24, 2020

Abstract

The vet uses the ultrasound to obtain the signal data concerning the spatial variations in the intestine where the marble is suspected to be. However, the data becomes very noisy due to the internal fluid movement. In order to locate the marble's trajectory, we have to first filter the noisy data. Using the Fast Fourier Transform to analyse the data in the frequency domain and through averaging theorem, we can obtain the center frequency, which will further help to filter the data around the center frequency. With these data analysis techniques, we can ultimately locate the marble location and destroy it to save the dog's life.

I. Introduction and Overview

Suppose a dog accidentally swallowed a marble into its intestine. Using ultrasound, people would obtain the data concerning the spatial variations in the intestine. However, the troublesome part is that the dog kept moving and the interior of intestine was fluid, which has generated much noise to the data. In order to locate and compute the accurate trajectory of the marble, we have to first filter the data. The use of Gaussian filter need us to first determine the frequency signature (center frequency) which can be obtained through averaging the spectrum. With the filtered data, we can determine the path of the marble and further the position of the marble at the 20th measurements. With this information, the vet can use the intense acoustic wave to precisely breakup the marble and save the dog's life.

II. Theoretical Background

Fourier Transform

Fourier Transforms are one of the most powerful techniques for time-frequency analysis. Formally, the transform for a function f(x) is an integral transform over the entire real line $x \in [-\infty, \infty]$ The Fourier Transform is

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

where the k represents wavenumbers. The transform converts the function of time to the function of frequency. The kernel of the transform, exp(+-ikx) determines the oscillatory behavior. Different from Continuous Fourier Transform, the Discrete Fourier Transform (DFT) works with finite sequence of equally-spaced samples of a function. Usually, we will compute the Fourier Transform on a finite domain $x \in [-L, L]$, which needs DFT. The inverse of Fourier Transform is

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

which converts the function on frequency domain to back its original time domain.
The Fast Fourier transform algorithm was developed to quickly perform the forward and backward Fourier transforms. FFT has the lowest operation count: O(N logN). Developed by Cooley and Tukey, the FFT's operation count grows only linearly like N even when N is huge. In order to apply the FFT algorithm, the range [-L, L] should be discretized into 2^n points. Usually, FFT algorithm has relatively good accuracy.

Filter Method

In real world application, the obtained digital signal data is often accompanied with high noise. With the help of FFT, we could treat the signal in its frequency domain to denoise the data. Filtering method could largely improve the data and extract the signal field under the noise. One of the filter method to consider here is Gaussian Filter:

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

where τ measures the bandwidth of the filter, and k is the wavenumber, k_0 is the center frequency where the highest frequency occurs. The filter function acts as a low-pass filter since it eliminates high-frequency components in the system of interest. By multiplying the filter function to the transformed function on the frequency domain, the frequencies

away from center frequency are strongly attenuated. The filtering process will reproduce an approximation to the signal field without much noise. By Inverse FFT, we could transform the filtered signal over frequency domain back to time domain. However, the use of filter function need us to determine the center frequency first.
Suppose the observed noise are white noise. Then the noise have a mean of zero. That is, the white noise are suppose to average to zero over the multiple(20) data measurements. Therefore, through averaging the 20 signals, we would obtain the data with approximate zero noise and further locate the center frequency.

III. Algorithm Implementation and Development

The implementation is divided into three parts:
1. Through averaging of the spectrum, determine the center frequency
2. Filter the data around the center frequency to denoise the data and obtain the path of marble
3. Compute the position of the marble at the 20th measurement
Note that the dataset contains 20 rows for 20 different measurements. The size 20×262144 implies that each row of record contains $64\times64\times64$ spatial data (in x,y,z directions) that are collected by ultrasound. That is, the time has three components in x,y,z direction. For each direction, the interval [-L, L] can be discretized into $64 = 2^6$ points in order to apply FFT. Additionally, the frequency k would also has three components in x,y,z direction. As the original Fourier Transform assumes 2π as the period, our k has to be rescaled to $2\pi/2L$. (Please see the Appendix B for more details)

1. Average the spectrum to determine the center frequency

To compute the average of the multiple signals, I first construct a zero `Uavg` with the size $64 \times 64 \times 64$. During each loop, each row of record is extracted through `reshape()` and performed multidimensional Fast Fourier Transform through `fftn`. The `Uavg` keeps adding up by each loop. Ultimately, the `Uavg` becomes the sum of all 20 signals and the white noise decreases to approximate zero. Dividing the absolute value of `Uavg` by 20, we essentially obtain the averaged result. From there, I extract the linear index where the maximum frequency occurs by `max()` and further obtain the multidimensional subscripts corresponding to the index, and the center frequency components in x, y, z directions.

2. Filter the data to obtain the marble path

After constructing the Gaussian filter function based on the center frequency obtained above, I perform the FFT on each sets of measurements. From there, I filter them by the specific function: $exp(-0.2 * ((Kx - kxc).^2 + (Ky - kyc).^2 + (Kz - kzc).^2))$ where the kxc, kyc and kzc are center frequency components in three directions. Then I reconstruct the filtered signal over time domain. Since the marble is always represented as maximum frequency, I extract the linear index where the maximum frequency occurs by `max()` and further obtain the multidimensional subscripts corresponding to the index. Based on these index, the position of marble in x, y, z direction is obtained during each measurements. To store this information, I create three vectors with a length 20: px, py, pz. Acknowledging the positions during the 20 sets of measurements, the marble path can be accurately constructed through `plot3()`.

3. Locate the marble at the 20th measurement

After obtaining the marble path, the position (x, y, z) at the 20th measurement can be computed by px(20), py(20), pz(20).

IV. Computational Results

Figure 1 shows the original noisy signal data at the 1th measurements. Through averaging process, the center frequency is computed to be [kxc kyc kzc] = (1.8850, -1.0472, 0). By iteratively filter each sets of measurements over frequency domain and reconstruct them over time domain, we obtain the marble position at each realization and further construct the marble path, shown as Figure 2. The final position of the marble is (-5.6250, 4.2188, -6.0938), labelled as the red circle on the path.

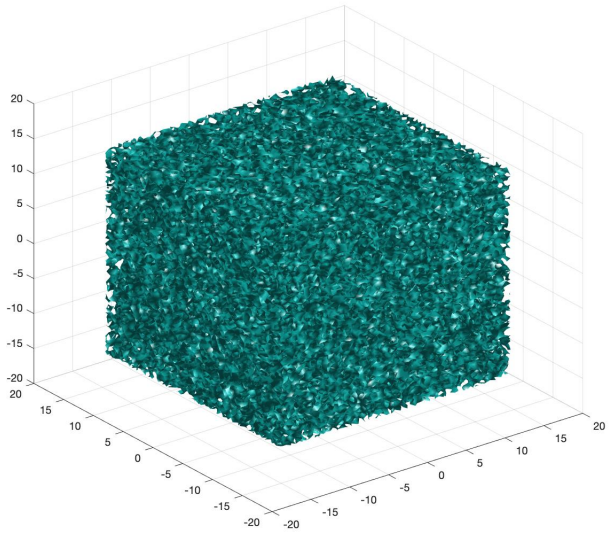

Figure 1: First Row Data Before Filtering

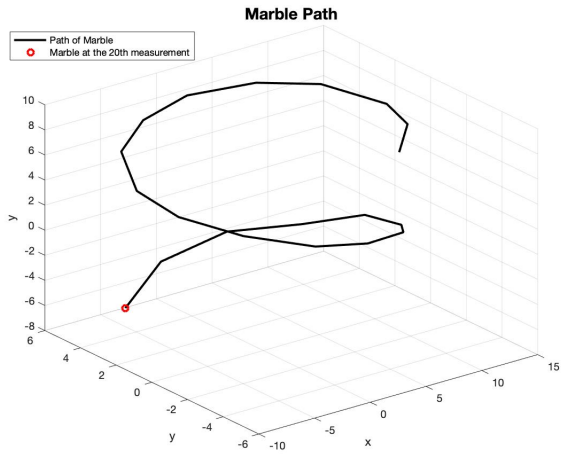

Figure 2: Path of the Marble

V. Summary and Conclusions

The purpose of this study is to find a way to determine the marble path and the position at the 20th measurement. Therefore the vet can employ intense acoustic wave to destroy the marble. Using the ultrasound, we could obtain the spatial data describing the movement of the marble. However, the high noise accompanied with the data makes our measurements obscure. To filter the signal, FFT transform is used to analyze the data on

frequency domain. Specifically, we would use the total 20 measurements to determine the frequency signature through averaging method. By computation, the center frequency is [kxc kyc kzc] = (1.8850, -1.0472, 0). Then we could apply Gaussian Filter to denoise the data on its frequency domain and reconstruct the function/data over time domain. The location of the maximum frequency during each measurement can be considered as the position of marble. With the 20 sets of measurements, we therefore construct the marble path and locate the marble at the 20th realization. Finally, the marble could be destroyed at (-5.6250, 4.2188, -6.0938) at the 20th measurement and the dog would be saved.

Appendix A MATLAB functions used and brief implementation explanation

`meshgrid:[X,Y,Z] = meshgrid(x,y,z)` returns 3-D grid coordinates defined by vectors x, y, z.
`reshape:Un(:,:,:)=reshape(Undata(1,:),n,n,n)` reshapes Undata(1,:) into a n by n by n arrays.
`fftn:fftn(Un)` N-D fast Fourier transform on Un
`ifftn()` inverse N-D FFT on ifftshift(Unft)
`ind2sub:[xi,yi,zi] = ind2sub(size(Uavg),idx)` returns a 3-array xi, yi, zi containing the equivalent multidimensional subscripts corresponding to the linear indices index for the Uavg object
`isosurface:isosurface(X,Y,Z,abs(Un),0.4)` computes isosurface data from the volume data abs(Un)
`plot3()` plot in 3D

Appendix B MATLAB codes

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

% first row data with noise
Un(:,:,:)=reshape(Undata(1,:),n,n,n);
close all, isosurface(X,Y,Z,abs(Un),0.4)
axis([-20 20 -20 20 -20 20]), grid on, drawnow
title('First Row of Data (with noise)','Fontsize',15);

% determine the center frequency by averaging 20 signals
Uavg = zeros(n,n,n);
for j=1:20
    Un(:,:,:) = reshape(Undata(j,:),n,n,n);
    Unt = fftn(Un);
    Uavg = Uavg+Unt;
end
Uavg = abs(Uavg)/20;
[value, idx] = max(Uavg(:));
[xi,yi,zi] = ind2sub(size(Uavg),idx);  % Convert linear indices to subscripts
kxc = Kx(xi,yi,zi);  % center frequency component in x direction
kyc = Ky(xi,yi,zi);  % center frequency component in y direction
kzc = Kz(xi,yi,zi);  % center frequency component in z direction

% filter the data to denoise
filter=exp(-0.2*((Kx-kxc).^2+(Ky-kyc).^2+(Kz-kzc).^2));
for j=1:20
    Un(:,:,:) = reshape(Undata(j,:),n,n,n);
    Unt = fftn(Un);
    Unft = Unt.*filter;
    Unf = ifftn(Unft);
    [value2,idx2] = max(Unf(:));    % idx2: the position in jth realization
    [xj,yj,zj] = ind2sub(size(Unf), idx2);
    px(j) = X(xj,yj,zj);  % position of marble in x direction for jth realization
    py(j) = Y(xj,yj,zj);
```

```
    pz(j) = Z(xj,yj,zj);
end

% plot the path of the marble, i.e the position during each realizations
plot3(px,py,pz,'k','Linewidth', 2, 'DisplayName', 'Path of Marble')
hold on

% the position of the marble at the 20th realization
plot3(px(20),py(20),pz(20),'ro', 'Linewidth', 2, 'DisplayName',
'Marble at the 20th measurement')
legend('location','best');
title('Marble Path','Fontsize',15);
xlabel('x'),ylabel('y'),zlabel('y'), grid on
hold off

% final position is
[px(20),py(20),pz(20)]
```

Reference

MathWorks: https://www.mathworks.com/help/matlab/ref/
J. Nathan Kutz. (2013) Data-Driven Modeling and Scientific Computation: Methods for Complex Systems and Big Data. Oxford University Press. ISBN:978-0-19-966034-6