Application of Fourier Transform in ultrasound

Author : Richy Yu

Abstract

This paper addresses a real life problem with Fourier Transform. The noisy data from the ultrasound is given, the purpose is to locate the object after denoising. The techniques used in this paper are denoising by averaging and filtering, specifically gaussian filter. It also involves visualization of the data. The programming software used is MATLAB.

Introduction and Overview:

The problem being addressed in this paper is the following. Your dog fluffy swallowed a marble. The vet suspects that it has now worked its way into the intestines. Using ultrasound, data is obtained concerning the spatial variations in a small area of the intestines where the marble is suspected to be. Unfortunately, fluffy keeps moving and the internal fluid movement through the intestines generates highly noisy data. The data contains 20 rows of data for 20 different measurements that were taken in time. The goal is to denoise the data and find the trajectory of the marble.

Firstly, I would apply the Fourier Transform and represent the data in frequency domain. Since I have 20 measurements, and I assume the noise is white noise, I can add up those measurements in the frequency domain, so that the noise could be canceled. Then I could determine the central frequency that the marble gives.

Secondly, I would use the central frequency to apply a gaussian filter to the data in the frequency domain. The filter will amplify the data at central frequency and reduce the noise at other frequencies. Then I take the Inverse Fourier Transform for the cleaned data, and hopefully I should be able to see the position of the marble.

Thirdly, I would repeat the second step for each measurement, plot the path of the marble, and determine the final position of the marble.

Theoretical Background

The data consists of discrete data points. Therefore, I will use Discrete Fourier Transform, which is defined as

$$\widehat{x_k} = \sum_{n=0}^{N-1} x_n e^{\frac{-2\pi ikn}{N}} \qquad (1)$$

$$\widehat{x_n} = \frac{1}{N}\sum_{n=0}^{N-1} \widehat{x_k} e^{\frac{2\pi ikn}{N}} \quad , \qquad (2)$$

where N is the total number. However, I would not use these two equations, because MATLAB has built-in function called 'fft', which does the transform for me. The idea of Fourier Transform is that it gives you a new function, which represents the original function as the strength at each frequency in the frequency domain. However, if the data has noise in it, the frequency that the marble gives cannot be determined easily. As a result, I will need to denoise the data.

The first technique of denoising the data is averaging. The condition of averaging is that the noise has to be white noise, meaning its affects are equal at different frequencies, and the noise has a mean of 0. This is the reason why the technique of averaging will work. If 20 different measurements are added, the noises should cancel themselves, because they have a mean of 0. As a result, a clean data is obtained in the frequency domain after averaging. One drawback of the technique is that after adding up the data in the frequency domain to cancel the noise, information in the original spatial domain is lost, which tells the position of the marble. So the purpose of averaging is to find the central frequency at which the function has the highest amplitude, and I can apply that central frequency to filter the data.

The second technique of denoising the data is filtering. In this paper, a gaussian filter is used, because it is common and easy. The condition of filtering is that we need a knowledgeable guess of frequency. A gaussian function is defined as centered at that central frequency and is multiplied with the data in the frequency domain, so that the data value at the central frequency will be amplified, and the noise at other frequencies will be reduced.

Therefore, a cleaned data can be obtained, and the information in the original domain is not lost.

Algorithm Implementation and Development

First, spatial domain is initialized as L = 15, and Fourier modes is defined as n = 64. We use the linspace(-L, L, n+1) command to generate 65 equally spaced points from -L to L, but we only pick the first 64 points, because the last point is just the opposite of the first point, and we are considering a periodic function. As a result, the last point can be thought of as the beginning of the next period. We are scaling the frequency so that the frequency values match up with the values in the spatial domain. Normally, MATLAB considers the function from -2π to 2π. We are dividing this frequency by 2L to rescale the frequency values.

The data has 20 rows, and each row represents one measurement. So we loop through each row, and use the function fft to take Fourier Transform of the data, and we add up those values in the frequency domain. I also normalized the average to that all values are less than one. Then I use the function isosurface to get a sense of where the central frequency could be (The graph is not shown). I computed the coordinates of frequency which give the maximum value of the function. This triple of coordinates is the central frequency of the data.

Then I define the gaussian filter in three dimensions. I loop through each row of the data again, and take the Fourier Transform of the data. Because of aliasing, I use the function fftshift to shift the data, so that it has the correct order, and I multiply it with the filter. Then I shift the data back and take the inverse Fourier Transform. Then I find the coordinates that give the maximum of the function. I tried several different values of bandwidth of the filter. By trials and errors, I believe 1 is a good value for bandwidth, because it gives a nice looking graph. This triple of coordinates is the position of the marble at that time of measurement. I repeat the process 20 times, so I get 20 positions of the marble and plot the trajectory of the marble. I plot the 20 positions on a single graph, as shown in Figure 1.

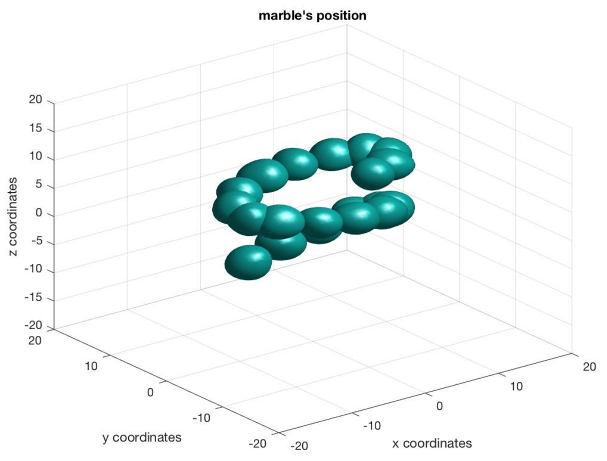

Figure 1: It shows the position of the marble at 20 different measurements. This figure is generated by isosurface.

Computational Results

The central frequency I got by averaging is (1.885, -1.0472, 0). After the filtering for twenty different measurements, I plot the positions of the marble on a single graph, as shown in figure 1. After calculating the exact positions of the marble at each measurement, I plot the path of the marble, as in figure 2. At 20th measurement, the marble's position is (-5.625, 4.2188, -6.0938), so the intense acoustic wave should be focused at this position to break the marble.

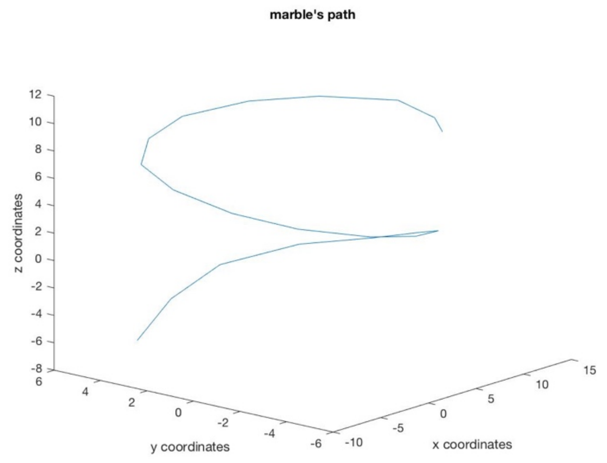

Figure 2 : It shows the trajectory of the marble at 20 different measurements. This figure is generated by plot3.

Summary and Conclusion :

This paper addresses the issue of noise data with the technique of averaging and filtering. I begin by taking the Fourier Transform of the data. By averaging in the frequency domain, central frequency can be determined, and this central frequency is used to apply a filter to the data at each measurement. Then I take the Inverse Fourier Transform to get the cleaned data in the original spatial domain and determine the location. The technique of averaging works if the noise is white noise, and multiple measurements are available. On the other hand, filtering requires a central frequency at which a filter is applied. These two techniques combined give an accurate position of the marble and satisfying result at the end.

Appendix A (MATLAB functions used) :

fftn : gives the Fourier Transform to the given input. Dimension is the same as input.
fftshift : shifts the input, so that the order matches the original order.
ifftn : undoes what fftn does.
ifftshift :undoes what fftshift does.
meshgrid : creates a grid of points
isosurface : plots contour plot in three dimensions.
plot3 : 3 dimensions plot

Appendix B (MATLAB codes) :

```
%%
clear; close all; clc;
load Testdata
L=15; % spatial domain
n=64; % Fourier modes
average = zeros(64, 64, 64);
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);
for j=1:20
   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
   ft = fftn(Un);
   average = average + ft;
end
average = abs(fftshift(average)) / max(max(max(abs(average))));
close all, isosurface(Kx,Ky,Kz, average, 0.9)
axis([-20 20 -20 20 -20 20]), grid on, drawnow
[M, I] = max(average(:));


kx0 = Kx(I);
ky0 = Ky(I);
kz0 = Kz(I);

%%
% Define Gaussian Filter

filter = exp(-((Kx - kx0).^2 + (Ky - ky0).^2 + (Kz - kz0).^2));
px = zeros(20);
py = zeros(20);
pz = zeros(20);

for i = 1 : 20
    Un(:, :, :) = reshape(Undata(i, :), n, n, n);
    Un_fft = fftn(Un);
    fftFilter = fftshift(Un_fft) .* filter;
    fftFilter = ifftshift(fftFilter);
    position = ifftn(fftFilter);
    position = position / max(max(max(position)));
    [M, I] = max(position(:));
    px(i) = X(I);
    py(i) = Y(I);
    pz(i) = Z(I);
    isosurface(X,Y,Z,abs(position),0.4)
    axis([-20 20 -20 20 -20 20]), grid on, drawnow
    hold on;
end
xlabel('x coordinates');
ylabel('y coordinates');
zlabel('z coordinates');
title("marble's position");
%%
close all;
plot3(px, py, pz);
xlabel('x coordinates');
ylabel('y coordinates');
zlabel('z coordinates');
title("marble's path");
```