HW1: An Ultrasound Problem

Eileen Chang

January 24, 2020

Abstract

In the real world, it is quite impossible to get clean raw data. Thus, remove the noise from the data will help people analyze the data more correctly. Fourier transform is a good tool to analyze the frequency of the signal and wave. We then will average the spectrum and find the frequency signature (center frequency) generated by the marble. Then, we implement the Gaussian Filter to eliminate the high frequency and keep the low frequency. Doing these will help us denoise the data and get the clearly answer to our problem.

1 Introduction

Suppose the dog, fluffy, swallowed the marble. And it is on the 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. However, fluffy kept moving, so the data has noise in it.

Due to the movement from fluid in fluffy's body, it would generate some random noise in the data. In order to denoise the data, we take the average of the frequency domain data to take out all the random frequencies in the data and consider only the frequency generated by the marble. The Gaussian filter can eliminate the frequency that was far away from the target frequency. Thus, we only consider the frequency that was close to the target frequency. Finally, we would convert the data back to the time domain to obtain the path of the marble. Thus, we would be able to know the coordinates that an intense acoustic wave should be focused to break up the marble at the 20th data measurement.

2 Theoretical Background

Fourier Transform and Gaussian Filtering are two mainly mathematical technologies that we used in the paper.

2.1 The Fourier Series

The Fourier series is a periodic functions that will break down the signal into different frequency, so it will not go forever. In this paper, we define the spatial

domain as $x \in [-L, L]$.

2.2 The Fourier Transform

The Fourier transform is a mathematical technique used to analyze the waveform (a signal or function). Its function and derivative can be represented by the sum of cosines and sines. It transforms a function of time, f(x), into a function of frequency, F(k). It is an integral transform defined over the entire line $x \in [-\infty, \infty]$. The Fourier transform is defined as

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

And its inverse, which converts the signals from the frequency domain to the time domain, is defined as

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

Once we are on a finite domain $x \in [-L, L]$, the continuous eigenfunction expansion becomes a discrete sum of eigenfunctions and eigenvalues. The Fourier transform doesn't provide any information about the time of these frequencies.

2.3 The Fast Fourier Transform

Since the Fourier transform performs a kind of slow, we use Fast Fourier Transform (FFT) in this paper. The FFT's operation count is O(Nlog(N)). Breaking the interval $x \in [-L, L]$ into 2^n points will lowering the operation to O(Nlog(N)). If the N is large, then the FFT will operate as linearly like N.

2.4 Gaussian Filtering And Time Averaging

Gaussian Filter is a tool that used to filter the noise in the signal. It is a low frequency filter, so it will eliminate the high frequency. This is the equation of the Gaussian Filter:

$$F(K) = e^{-\tau(K-K0)^2} \tag{3}$$

where τ measures the width of the filter and K0 is the frequency that we are interested in. These are high-frequencies in relation to the center-frequency (K = K0) of the desired signal field. Application of the filter attenuates strongly those frequencies away from center frequency K0. Thus, if there is a signal near K0, the filter isolates the signal input around this frequency.

Since the noise in data, we will average the spectrum over time. So, we can diminish the random data in order to get a clear path of the marble.

3 Algorithm Implementation and Development

In order to find the trajectory of the marble, we will go to implement the algorithm to find the final position of the marble. All the implementations are done in Matlab.

1. Load the testdata.mat into the Matlab.
2. Define the domain:
   - Spatial space (the time domain): $x \in [-L, L]$, L = 15
   - Fourier mode ($2^n = 64$)
   - Grid vector (evenly distribute the 2^n+1 points into [-L,L].
3. Set the frequency components by scaling the time domain with $\frac{2\pi}{2L}$, so the frequency domain will be $[-2\pi, 2\pi]$.
4. Swap the first half and second half of frequency components by using the commend fftshift, so the function will transform back to its mathematically correct positions.
5. Create a 3D cartesian grid by using the commend meshgrid.
6. Using the data in testdata.mat to calculate the average of the frequency.
   - Reshape the testdata.mat into the 3D ($64 \times 64 \times 64$).
   - Use Fourier Transform to transform the 3D data into the frequency domain by using the commend fftn.
   - Then shift the data back and take the absolute value of the complex number.
   - Getting the average by divide the time step, so we can get rid of the noise.
7. Find the indices (target frequency coordinate) in Fourier transform that has the largest value
8. Then, we filter the data by multiply the Gaussian filter to diminish the frequency that are far away from the center frequency.
9. Convert back to the time domain by using the commend ifftn.
   - The coordinate that has the maxi value in the spatial domain in the final step is the marble's final position.

4 Computational Results

After running the algorithm above, we get the final marble's position. The final coordinate of the marble is (x,y,z) = (-5.625, 4.2188,-6.0938).

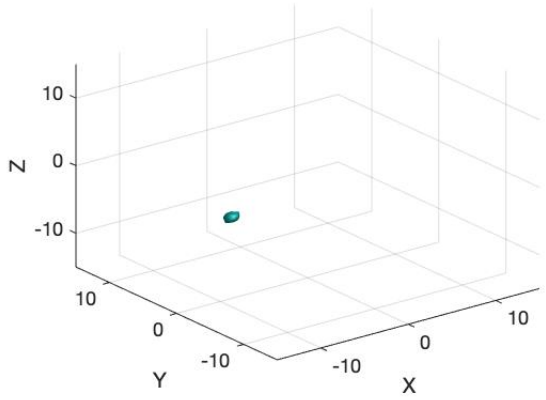
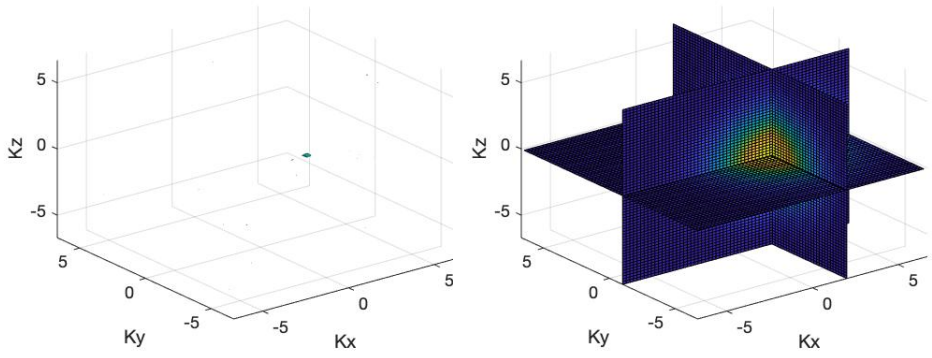


Figure 1: Isosurface of the final position of the marble

Figure 2(a) is the frequency distribution of the signal after averaging by the time steps. Figure 2(b) is the Gaussian function that applies to the target frequency point.

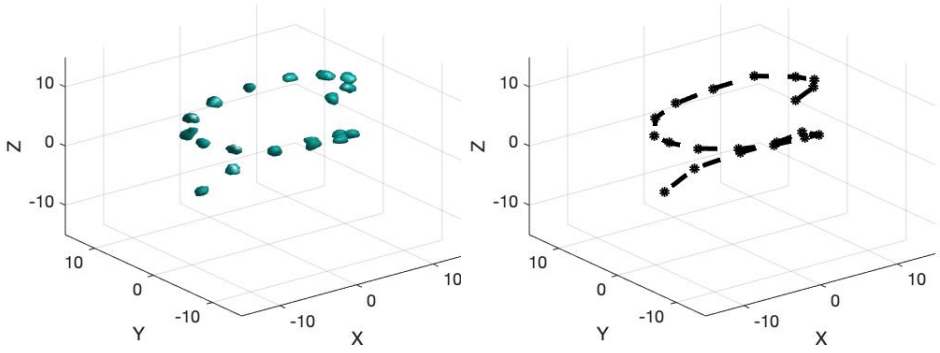


(a) Frequency Distribution of Signal (b) Gaussian Filter

Figure 2: Frequency Distribution Signal and Gaussian Filter

Figure 3(a) is the isosurface of the trajectory of the marble. After multiplying the Gaussian filter to the signal and take the inverse of the Fourier

transform, we then get this figure. Figure 3(b) is drawn by using the commend plot3. It also indicates the trajectory of the marble.



(a) Isosurface (b) Plot3

Figure 3: Isosurface and Plot3 for the trajectory of the marble

5 Summary and Conclusions

In order to save the fluffy's life, an intense acoustic wave should be focused to break up the marble at the 20th data measurement at $(x, y, z) = (-5.625, 4.2188, -6.0938)$.

Using the Fourier Transform, Gaussian Filter, and averaging of the spectrum let us be able to denoise the data and find out the marble's final position.

6 Appendix A.

1. linspace
   - Generate linearly spaced vector.
   - x2 = linspace(-L,L,n) will generate n points in the interval [-L,L].
2. meshgrid
   - 2-D and 3-D grids
   - $[X, Y, Z] = meshgrid(x, y, z)$ returns 3-D grid coordinates defined by the vectors x, y, and z.
3. zeros()
   - Generate a vector that only contains zero.
   - zeros(10,1) will generate the 10X1 vector that fill with zero.

4. max()
    - Find the maxi value in the vector/matrix.
    - M = max(A) will return the max value in array A.
5. abs()
    - Return an absolute value.
    - A = abs(a).
6. fftn()
    - N-D fast Fourier transform
    - Y = fftn(X) returns the multidimensional Fourier transform of an N-D array using a fast Fourier transform algorithm.
7. ifftn()
    - Multidimensional inverse fast Fourier transform
    - X = ifftn(Y) returns the multidimensional discrete inverse Fourier transform of an N-D array using a fast Fourier transform algorithm.
8. fftshift()
    - Shift zero-frequency component to center of spectrum
    - Y = fftshift(X) rearranges a Fourier transform X by shifting the zero-frequency component to the center of the array.
9. reshape()
    - Reshape the dimension of data
    - B = reshape(A,sz) reshapes A using the size vector, sz, to define size(B).
10. plot3()
    - 3-D point or line plot
    - plot3(X,Y,Z,LineSpec) creates the plot using the specified line style, marker, and color.
11. isosurface()
    - Extract isosurface data from volume data
    - isosurface(X,Y,Z,f,isovalue).

7 Appendix B.

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

L = 15; %spatial domain
n = 64; %Fourier modes

%Define grid vectors, create 3D cartesian grid
x2 = linspace(-L,L,n+1); % time discretization
x = x2(1:n); % only use the first n points (periodicity)
y = x;
z = x;
k = (2*pi/(2*L))*[0:(n/2-1) -n/2:-1]; % frequency
    components
ks = fftshift(k);

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

%the variable that going to calculate the average of
    frequency
U = zeros(n,n,n);

for j = 1:20
    Un(:,:,:) = reshape(Undata(j,:),n,n,n);
    U = U + fftn(Un);
end

%shift and take absoluate value of average frequency
U = abs(fftshift(U))/20;

%Finding the center frequency (max value in U)
maxi = -inf;
for a = 1:64
    for b = 1:64
        for c = 1:64
            if U(a,b,c) > maxi
                maxi = U(a,b,c);
                x1 = a;
                y1 = b;
                z1 = c;
            end
        end
    end
```

```
end


%Gaussian Filter Function
tau = 0.2;
gfunc = exp(-tau*((Kx-ks(y1)).^2 + (Ky-ks(x1)).^2 +(Kz-ks
    (z1)).^2)/2);

figure(1)
isosurface(Kx,Ky,Kz,abs(U)/max(U(:)),0.7)
set(gca,'FontSize', 18);
axis([ks(1) -ks(1) ks(1) -ks(1) ks(1) -ks(1)]), grid on;
xlabel('Kx');ylabel('Ky');zlabel('Kz');

%Gaussian Filter picture
figure(2)
slice(Kx,Ky,Kz,gfunc,2,-1,0);
set(gca,'FontSize', 18);
axis([ks(1) -ks(1) ks(1) -ks(1) ks(1) -ks(1)]), grid on;
xlabel('Kx');ylabel('Ky');zlabel('Kz');

marble_location = zeros(20,3);
for j = 1:20
    Un(:,:,:) = reshape(Undata(j,:),n,n,n);
    Ut = fftshift(fftn(Un));
    NewUt = Ut.*gfunc;
    Unf = abs(ifftn(fftshift(NewUt)));

    %find the indices that has max value
    maxi = -inf;
    for a = 1:64
        for b = 1:64
            for c = 1:64
                if Unf(a,b,c) > maxi
                    maxi = Unf(a,b,c);
                    x1 = X(1,a,1);
                    y1 = Y(b,1,1);
                    z1 = Z(1,1,c);
                end
            end
        end
    end

    marble_location(j,:) = [y1,x1,z1];

   %isosurface of trajectory of marble
```

```
    figure(3)
    isosurface(X,Y,Z,abs(Unf)/max(abs(Unf(:))),0.8);
    set(gca,'FontSize', 18);
    axis([-L L -L L -L L]), grid on;
    xlabel('X');ylabel('Y');zlabel('Z');
end

final_location = marble_location(end,:);

%isosurface of final marble position
figure(4)
isosurface(X,Y,Z,abs(Unf)/max(abs(Unf(:))),0.8);
set(gca,'FontSize', 18);
axis([-L L -L L -L L]), grid on;
xlabel('X');ylabel('Y');zlabel('Z');

%plot3 of the trajectory of the marble
figure(5)
plot3(marble_location(:,1), marble_location(:,2),
   marble_location(:,3),'k--o','LineWidth', 5);
set(gca,'FontSize', 18);
axis([-L L -L L -L L]), grid on;
xlabel('X');ylabel('Y');zlabel('Z');
```