Determine Central Frequency in Noisy Data

Yinxi Pan

AMATH482, Homework1, Winter2020

January 25, 2020

Abstract

This report describes methods to obtain the exact location of the marble inside a dog's intestines from an ultrasound dataset that contains 20 distinct measurements. The main methods used are called averaging and filtering which are based on the Fourier Transform. The frequency signature sent by the object is found by averaging the spectrum. The exact location of the object during each measurement is determined by filter out the data around the center frequency. The path of the marble is located by plotting the location of it in different measurements.

0.1 Introduction and Overview

There's a dog that swallowed a marble and the vet suspects the marble has now been in the intestines of it by taking the ultrasound for 20 times. To save the dog, we need to find the path and exact location of the marble and break it down. However, the dog keeps moving and the internal fluid movement through the intestines generates highly noisy data. To find the exact location of the marble, we need to denoise the data with certain methods. This report focuses on the method of averaging and filtering which relies on the Fourier Transform. Fourier Transform also called spectral transform is one of the most powerful and efficient techniques for solving a wide variety of problems arising from the physical, biological and engineering sciences. [1] The idea of Fourier Transform is to represent some function $f(x)$ by $\cos(kt), \sin(kt)$, which is similar to take a signal and break it down to different frequencies. This property helps us to realize the transformation between the spatial domain and frequency domain. The report includes all the theoretical backgrounds of the methods mentioned above, how we developed the algorithm using those methods to save the dog's life and at last appended with the MATLAB script.

[1] Kutz, J. Nathan. Data-Driven Modeling Scientific Computation: Methods for Complex Systems Big Data, Chap13. OUP Oxford, 2013.

0.2 Theoretical Background

0.2.1 Fourier Transform

Fourier Transform(FT), as mentioned above, is a technique to rewrite some functions into sum of $\sin(kt), \cos(kt)$. By Euler's formula(equation(1)), we could further rewrite the transformed equation as equation(2):

$$e^{i\theta} = \cos\theta + i\sin\theta, i = \sqrt{-1} \tag{1}$$

$$\hat{f}(k) = \frac{1}{\sqrt{2\pi}} \int_{-\infty}^{\infty} f(x) \cdot e^{-ikx} dx, i = \sqrt{-1}, x \in [-\infty, \infty] \tag{2}$$

and the inverse Fourier Transform is defined to be as in equation(3):

$$f(x) = \frac{1}{\sqrt{2\pi}} \int_{-\infty}^{\infty} \hat{f}(k) \cdot e^{ikx} dk, i = \sqrt{-1}, k \in [-\infty, \infty] \tag{3}$$

Fourier Transform also works for discrete numbers. If $\{X_0, \cdots, X_{N-1}\}$ are all the points we have, N points in total, then the Discrete Fourier Transform is defined as in equation(4):

$$\hat{X}_k = \sum_{n=0}^{N-1} X_n e^{\frac{-2\pi ikn}{N}}, k = 0, 1, \cdots, N-1 \tag{4}$$

where $\frac{N}{2}$ is the highest frequency. The inverse discrete transform is similar to the continuous case with one more coefficient as $\frac{1}{N}$.

0.2.2 Fast Fourier Transformation

Fast Fourier Transformation(FFT) is an implemented algorithm in MATLAB developed to perform backward and forward Fourier Transform. We could use the built-in command in MATLAB, for example *fft(),fftn()* [2] , instead of actually computing the transformations. There are two major properties of FFT. The first one is the FFT transforms data on an interval $x \in [-L, L]$. The FT we introduced in the previous part takes e^{ikx} inside by Euler's formula which is oscillatory. Then we need periodic boundary conditions to apply FFT. The second one is we need to discretize the range of x into 2^n points such that the number of points should be 2,4,8, $\cdots$. The reason for this is because FFT is designed to have operation counts of O(N log N), that's also why it's called fast. To achieve this operation counts mathematically we need to have points in 2^n.

0.2.3 Method of Averaging

The ultrasound/radar data we obtained are integrated with a large amount of noise. This kind of noise is called the white noise which is a noise that affects all frequencies the same [3]. One property of the white noises is that they are normally distributed with $\mu = 0$ and

[2] MATLAB functions explanations please see Appendix.A

[3] Kutz, J. Nathan. Data-Driven Modeling Scientific Computation: Methods for Complex Systems Big Data, Chap13.2. OUP Oxford, 2013.

$\sigma = 1$. With this property, by adding up all the frequency together and dividing the number of measurements we have for the frequency, most white noises will cancel out with each other and leave a more distinguishable signal which is the center frequency to us. The method of adding up all frequency and then dividing to the average is called the method of averaging.

0.2.4 Method of Filter

Spectral filtering is a method which allows us to extract information at specic frequencies [4]. Applying the filter to a dataset means to multiply the filter with the dataset. Here we introduce one of the simplest filters which is called the Gaussian filter as in equation(5)

$$\mathcal{F}(k_1, k_2, \cdots, k_3) = e^{-\tau((k_1-k_{10})^2+(k_2-k_{20})^2+\cdots+(k_n-k_{n0})^2)} \tag{5}$$

where τ measures the width of the filter, k is all the frequency we detected and k_0 is the center frequency in n-dimension. The filter is called Gaussian since it's normally distributed. Multiplying the filter with the frequency data, we can reduce the noise around the center frequency because the normal distribution goes to 0 when it's far away from the center.

0.3 Algorithm Implementation and Development

1. Load the data: 'Testdata.mat' and create axes for spatial and frequency domain. [5]
   (a) Since we are going to use *fftn()* [6] later, we need to discritize the range of x into 2^n. we discretize the range into 64 points since the size of the data is 64*64*64.
   (b) For the frequency domain, we need to rescale all frequencies by $\frac{2\pi}{L}$ since the *fftn()* assumes 2π periodic signals.
   (c) For frequency domain, we need to use *fftshift()* [7] to make the center frequency later located on the middle of the sequence for plotting(avoid aliasing phenomena happen).
2. [8]Apply method of averaging
   (a) Create a loop for 20 times of averaging (measurements in total):
      i. Reshape the ultrasound data into spatial 3D by *reshape()*[9]
      ii. Apply *fftn()* on the spatial data and add up the frequency.
   (b) Average the frequency. Apply *fftshift()* again to avoid aliasing phenomena and take absolute value of the average since *fftn()* generates imaginary numbers.

[4]Kutz, J. Nathan. Data-Driven Modeling Scientific Computation: Methods for Complex Systems Big Data, Chap13.2. OUP Oxford, 2013.
[5]MATLAB Code please see Appendix.B line1-20
[6]MATLAB functions explanations please see Appendix.A
[7]MATLAB functions explanations please see Appendix.A
[8]MATLAB Code please see Appendix.B line22-42
[9]MATLAB functions explanations please see Appendix.A

(c) Use *isosurface()*[10] to plot the averaged frequency data in frequency domain.

(d) Get the center frequency by the max command[11].

3. [12]Apply method of filtering

   (a) Create the Gaussian filter, where the center frequency (kx_0, ky_0, kz_0) (3-dimension as defined by equation(4)) is what we determined above.

   (b) Shift the value generated by the filter function to avoid aliasing again.

   (c) Create 3 vectors to store the x, y, z coordinates in spatial domain and create a loop for 20 times of filtering:

      i. Reshape the ultrasound data into spatial 3D.

      ii. Apply FFT on the spatial data and get the location of center frequency by max command.

      iii. Apply the Gaussian filter on the frequency domain.

      iv. Do the inverse FFT on filtered data to get the position in spatial domain. We take the absolute value again to get real numbers only.

      v. Record the position in spatial domain. The location indices are generated by the max command used before.

4. Use *plot3()*[13] to plot x, y, z coordinates. The last elements in spatial coordinate vectors form the final location of the marble.

0.4 Computational Results

Running the MATLAB script in Appendix.B, we obtain two figures as Figure(1) and Figure(3). Figure(3) is the isosurface visualization in the frequency domain with isovalue 0.999 after applying the average method, where the most distinguishable dot is the center frequency. The center frequency is at $(1.8850, -1.0472, 0)$ in the frequency domain. After filtering, we record the location of the marble at each time and then obtain the plot of its path as in Figure(1). The final position of the marble is at $(-5.6250, 4.2188, -6.0938)$ in the spatial domain. Also, if we un-comment the MATLAB code inside the averaging loop, we get some impressions of how the original data looks like as in Figure(2) which is pretty messy.

0.5 Summary and Conclusions

To save a dog's life, who swallowed a marble, we take the ultrasound for 20 times. Based on the ultrasound data, we determined the path(Figure(1)) of the marble, center frequency generated by the marble ($(1.8850, -1.0472, 0)$ in frequency domain) and final location of the

[10]MATLAB functions explanations please see Appendix.A

[11]MATLAB functions explanations please see Appendix.A

[12]MATLAB Code please see Appendix.B line62-84

[13]MATLAB functions explanations please see Appendix.A

marble($(-5.6250, 4.2188, -6.0938)$ in spatial domain) inside the dog by using the method of averaging and filtering based on FFT in MATLAB. Averaging diminishes most white noises and filtering obtains the path and exact location of the marble. With the information from these methods, We could apply an intense acoustic wave on the marble at the final location to break it up and save the dog's life.

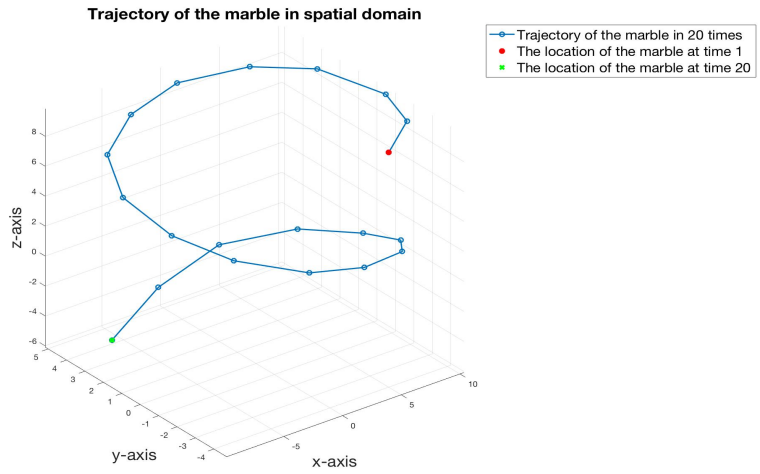

Figure 1: The trajectory of the marble inside the dog in spatial domain for 20 measurements, which is convincing since we know the marble is inside the intestines which looks like the spiral as in the gure. The red dot is the position at time1 and the green dot is the position at time20. The center frequency is at $(1.8850, -1.0472, 0)$ with the width of the filter = 1.

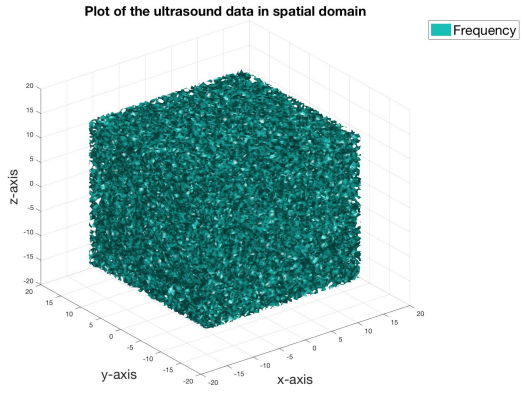

Figure 2: The illustration of the original dataset by *isosurface()* with isovalue = 0.4

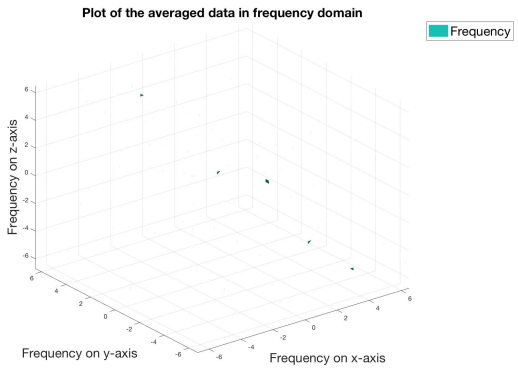

Figure 3: Averaged frequency data plotted by *isosurface()* with isovalue = 0.999. Isovalue determines the size of the detected signal. When plotting, we normalize the size of the averaged frequency so that it's easier to find the best isovalue.

Appendix A. MATLAB functions used[14]

1. Ks = fftshift(X): This function rearranges a Fourier transform X by shifting the zero-frequency component to the center of the array. This function avoids the aliasing when plotting the Fourier Transform v.s. the frequency.

2. $[X, Y, Z]$ = meshgrid(x,y,z): This function returns 3-D grid coordinates defined by the vectors x, y, and z. The grid represented by X, Y, and Z has size length(y)-by-length(x)-by-length(z).

3. Un(:,:,:) = reshape(Undata(j,:),n,n,n): This function reshapes the j^{th} column of Undata into a n-by-n-by-n array where n is the size of each dimension.

4. Y = fftn(X) : This function returns the multidimensional Fourier Transform on X. Y as the output should have the same size as X. This is expanding the fft() which applies to 1-dimensional Fourier Transform.

5. Y = ifftn(X): This function returns the multidimensional discrete inverse Fourier transform of an N-D array using a fast Fourier transform algorithm. Y as the output should have the same size as X. This is expanding the ifft() which applies to 1-dimensional Fourier Transform.

6. isosurface(X,Y,Z,V,isovalue) : This function computes isosurface data from the volume data V(V must be a vector) at the isosurface value specified in isovalue. That is, the isosurface connects points that have the specified value much the way contour lines connect points of equal elevation. We should always change the isovalue to observe the data.

7. plot3(X,Y,Z): This function plots the X,Y,Z vector in 3-D space. If wee want the set of coordinates connected by line segments, we need to specify X, Y, and Z as vectors of the same length.

8. zeros(n,m): This function create a n-by-m matrix of 0s.

9. $[C, I]$ = max(v(:)): This function take in a vector or multi-dimension matrix and return the maximal value among all and the corresponding index for that value in v.

[14]The full description of each function below can be foun on : "MATLAB." MATLAB Documentation, www.mathworks.com/help/matlab/.

Appendix B. MATLAB codes

```
% load data
clear; close all; clc;
load Testdata

% define axes on both spatial and frequency domain
L = 15; % spatial domain
n = 64; % Fourier modes — frequency domain
x2 = linspace(-L, L, n + 1); % discretizing the range of x
% create the axis
x = x2(1 : n);
y = x;
z = x;
% rescale frequencies by 2*pi/L since the FFT assumes 2*pi periodic signals.
k = (2 * pi / (2 * L)) * [0 : (n / 2 - 1) -n / 2: -1];
% for plotting or the order is going to mess up
ks = fftshift(k);
% create coordinates defined by x,y,z - for spatial domain
[X, Y, Z] = meshgrid(x, y, z);
% create coordinates defined by x,y,z - for frequency domain
[Kx, Ky, Kz] = meshgrid(ks, ks, ks);

%% 1.Averaging the nosiy data
ave = 0;

for j = 1:20
    % reshape each time ultrasound into 3D spatial domain
    Un(:,:,:) = reshape(Undata(j,:),n,n,n);

    % Below the codes give an overview of the noisy data. Uncomment them and
    % plot the noisy data if you want.

    % close all, isosurface(X, Y, Z, abs(Un), 0.4)
    % axis([-20 20 -20 20 -20 20]), grid on, drawnow
    % pause(1)

    % apply FFT on each time
    Unt = fftn(Un);
    ave = ave + Unt;
end

% get ride of the imaginary numbers since we consider the spatial domain
ave = abs(fftshift(ave)) ./ 20;

figure(1)
% plot the averaged data
isosurface(Kx,Ky,Kz, ave ./ max(ave), 0.999)
axis tight % zoom into the area that contains data
xlabel('Frequency on x-axis', 'FontSize', 20)
ylabel('Frequency on y-axis', 'FontSize', 20)
```

```
zlabel('Frequency on z-axis', 'FontSize', 20)
title('Plot of the averaged data in frequency domain', 'FontSize', 20)
legend('Frequency', 'FontSize', 20)
grid on
drawnow

% find the max value of average and return the index of that position
[C,I] = max(ave(:));
center = [Kx(I), Ky(I), Kz(I)];
disp('The center frequency in frequency domain is at:')
center

%% 2. Guassian Filter
tau = 0.7;

g = exp(-tau * ((Kx - center(1)).^2 + (Ky - center(2)).^2 + (Kz - ...
    center(3)).^2));

g = fftshift(g);

Xc = zeros(1,20);
Yc = zeros(1,20);
Zc = zeros(1,20);

for j = 1:20
    Un(:,:,:)=reshape(Undata(j,:),n,n,n);
    Unt = fftn(Un);
    Unft = g .* Unt;
    Unf = ifftn(Unft);
    Unf = abs(Unf); % imaginary
    [C1,I1] = max(Unf(:));
    Xc(j) = X(I1);
    Yc(j) = Y(I1);
    Zc(j) = Z(I1);
end

disp('The final position in spatial domain is:')
[Xc(20), Yc(20), Zc(20)]

figure(2)
plot3(Xc,Yc,Zc,'-o','Linewidth', 1.6)
hold on
% time1 location
plot3(Xc(1), Yc(1), Zc(1), 'r*', 'Linewidth', 3)
% time20 location
plot3(Xc(20), Yc(20), Zc(20), 'gx', 'Linewidth', 3)
axis([-20 20 -20 20 -20 20])
xlabel('x-axis', 'FontSize', 20)
ylabel('y-axis', 'FontSize', 20)
zlabel('z-axis', 'FontSize', 20)
title('Trajectory of the marble in spatial domain', 'FontSize', 20)
legend('Trajectory of the marble in 20 times', 'The location of the marble ...
    at time 1', 'The location of the marble at time 20', 'FontSize', 17)
```

```
101 axis tight % zoom in data
102 grid on
103 drawnow
```