AMATH 482 Homework 1

By Dexian Chen

Abstract

The report intends to provide explanations and visualizations of a process to manipulate data generated by ultrasound in a dog's intestine. We apply a series of techniques such as Fast Fourier Transform, Averaging, Gaussian Filter, to approximate the location of the target.

Introduction and Overview

A dog Fluffy swallowed a marble, so the vet utilized an ultrasound to obtain a set of data concerning the spatial variations in a small area of the intestines, where the marble is suspected to be. Since Fluffy keeps moving, the data is noisy. Given this set of data, our mission is to denoise the data to get the location of the marble for breaking the marble with intense acoustic wave to save Fluffy.

Theoretical Background

A famous Mathematician Fourier discovered that any periodic function including discontinuous ones defined over real numbers can be written as a combination of sin and cos. Fourier Series is defined as

$$f(x) = \frac{a_0}{2} + \sum_{n=1}^{\infty} (a_n \cos nx + b_n \sin nx), \qquad x \in (-\pi, \pi]. \qquad (1)$$

We have this expansion $x \in (-\pi, \pi]$ because $f(x)$ is constructed from a sine and cosines which are all in a 2 π period.

Sometimes, people will have to deal with non-periodic functions instead of perfect functions, so scholars extend the Fourier Series to Fourier Transform for non-periodic functions. The trick of derivation is thinking non-periodic function as periodic one with infinite period. Fourier Transform and its inverse are defined as:

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

$$f(x) = \frac{1}{\sqrt{2\pi}} \int_{-\infty}^{\infty} e^{-ikx} F(k) dk \quad (3)$$

Understanding that the Fourier Transform allows us to decompose and transform a signal function in time/spatial domain to its component frequency is significant in this project. Inverse Fourier Transform is just an inverse process of Fourier Transform. Fast Fourier Transform is the improved version of Fourier Transform with operation count of O(N log(N)). When filtering the noise, we use the general Gaussian filter defined as:

$$\mathrm{F}(x) = \exp(-\tau * (\mathrm{k} - k_0)^2) \quad (4)$$

where τ is the bandwidth of the filter, k stands for frequency, and k_0 stands for the center frequency. We apply this filter by simply multiply it with the object function. Our choice of τ effect how quickly the signal is denoised away from the center frequency.

Algorithm Implementation and Development

We choose our spatial domain L = 15. To apply Fast Fourier Transform (FFT) and Inverse Fast Fourier Transform (IFFT), we need to set our discretization of x to match the number of 2^n. Furthermore, we rescale our frequency k by 2π/L, because the FFT assumes 2π period, and set up the mesh grid by the command **meshgrid** for spatial domain and shifted frequency domain, which are [X Y Z] and [Kx Ky Kz]. (Note: Kx Ky and Kz are shifted order k by the command **shift**) After the set up, we start to take average of the data set to find center frequency. (See Appendix B Line) In an iterative process, I **reshape** each row of the data into 3D array with size of 64x64x64, and then apply the command **fftn** to transfer it from spatial domain to frequency domain. Finally, I add up all the 20 signals in frequency domain and find the average, denoted as "**ave**".

Before defining the filter around the center frequency, I find the 3D index of the center frequency by taking absolute value, normalizing it, and apply **max** and **ind2sub**, which allows me to find the linear maximum index and then shift it to subscripts. Hence, I can use the subscripts to find center frequency in [Kx Ky Kz] and build up the filter with it. (See Appendix B Line) Since I build my filter with shifted order k, which is [Kx Ky Kz], I to unshift my filter's order back to the original by **ifftshift**.(See Appendix B Line) My order is matched so far. Therefore, I begin filter each of my data by looping through all my 20 rows of measurement given to us. During the loop, I **reshape** each row of data to 64x64x64, apply **fftn** again to make each data in frequency domain, and then apply the filter by multiplication. Now, the signal data should be neat, so we are ready to find the marble locations in these 20 measurements (each row of data corresponds to 1 measurement).

To build efficient algorithm, I do the following immediately after filtering in the same loop above. I apply IFFT by **ifftn** to transfer filtered data to spatial domain find the marble coordinates with **max** and **ind2sub**, which is exactly the same process as finding the index of

center frequency. After that, I store each measurement's index to an 20x3 array called **marble_index**. Finally, I can plot the 20th marble and trajectory with the **plot3** and **isosurface** command. (See Appendix B Line)

Computational Results

After averaging, the exact center frequency index and value can be found as the following:

Index [28, 42, 33] in [Kx Ky Kz]

Center frequency k0 = [1.8850, -1.0472 0]

While storing the marble coordinates, I also use **isosurface** to visualize the 20 marble's positions in the loop.(See Figure 1) Finally, **plot3** is used to plot trajectory and the 20th marble(RED CIRCLE) for intense acoustic wave to focus on breaking the marble in the dog's intestine.(See Figure 2)

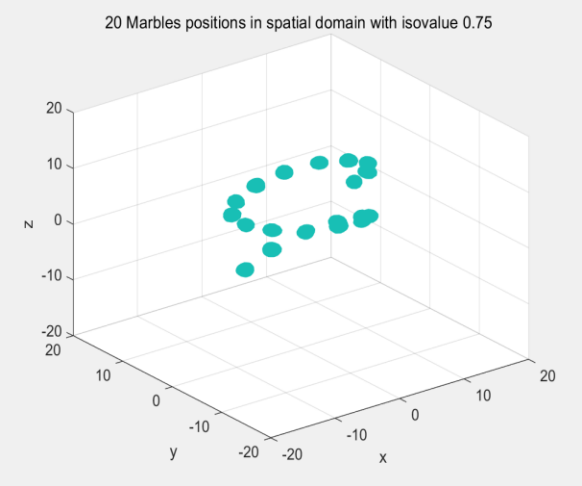

Figure 1

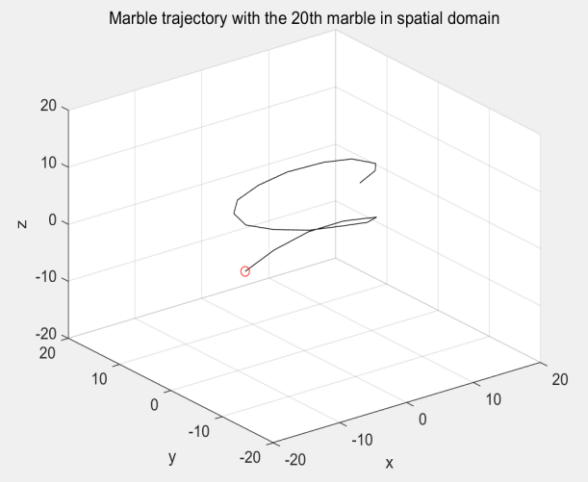

Figure 2

The exact position of the 20th marble is listed in the following visualized in next page, Figure 3:

20th marble coordinate: (-5.625, 4.2188, -6.0398)

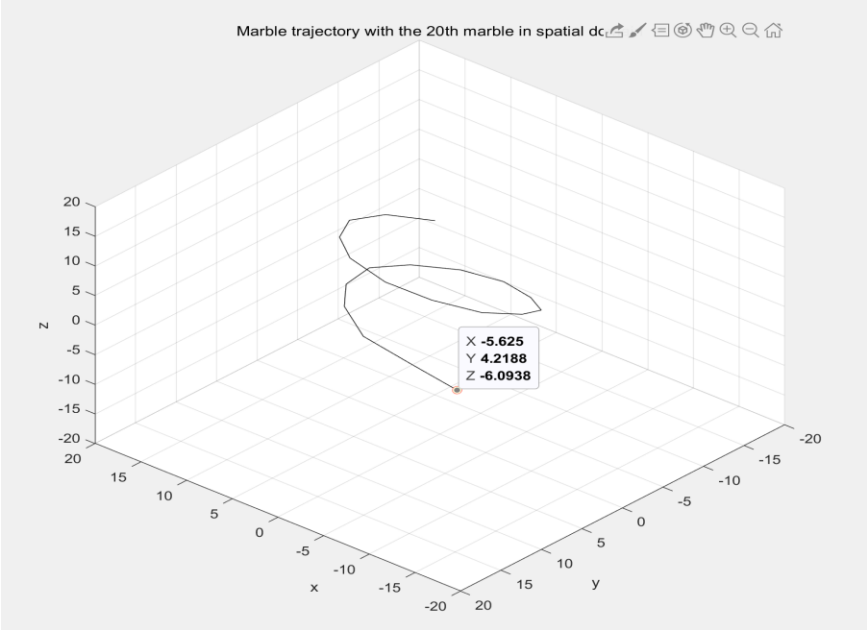

Figure 3: 20th Marble Coordinate

Summary and Conclusions

In conclusion, we take the following actions to locate the marbles through given signal data to save Fluffy: 1. We find the average of the signal data to cancel out white noises which have zero mean, so that we can find the center frequency-the maximum of averaged signal data 2. With center frequency, we can build up filter to apply to each row to clean the noise. Hence, we can eventually approximate the trajectory of the marbles and locate the 20th marble to help the vet remove it with intense acoustic wave. Averaging is significant mathematical techniques to find the center frequency. FFT and IFFT are the most important tools to switch between spatial domain and frequency domain for our analysis.

Appendix A

meshgrid: [X, Y, Z] = meshgrid(x,y,z) returns 3-D grid coordinates defined by the vectors x, y, and z.

reshape: reshape (A, a_1 , a_2 , … , a_n) reshape vector A and return it in a size of a_1 x a_2 x…x a_n.

isosurface: plot the isosurface of the input 3D array for a given 3D coordinate.

fftn(x): apply fast fourier transform in N dimension on x.

ifftn(x): apply inverse fast fourier transform in N dimension on x.

fftshift(x): rearranges a Fourier Transform X by shifting the zero-frequency component to the center of the array

max: [M, I] = max(input) as an example, the command max return the maximum index as I, and maximum index as M

ind2sub: convert linear index to subscripts indes.

Plot3(X, Y, Z): plot given 3D coordinates in 3 dimensional system.

Appendix B

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

% Visualize original data
figure(1)
title('Unfiltered signal in spatial domain')
for j=1:20
Un(:,:,:)=reshape(Undata(j,:),n,n,n);
isosurface(X,Y,Z,abs(Un),1)
axis([-20 20 -20 20 -20 20]), grid on
end
xlabel('x')
ylabel('y')
zlabel('z')

% Find the center frequency by averaging all signals
ave = zeros(n,n,n);
for i=1:20 % iterate through 20 measurements and find the total sum of the 20 measurement
Un(:,:,:)= fftn(reshape(Undata(i,:),n,n,n)); % reshape each measurement and swtich to frequency
domain by applying fftn
ave = ave + Un; % add up each measurement
end
ave = abs(fftshift(ave))./20;        % rearrange the order and find average
ave = ave./max(ave(:));              % normalize the average
[M I] = max(ave(:));                 % find maximum value and linear index
[x_ind, y_ind, z_ind] = ind2sub(size(ave),I); % convert linear index into subscripts to find center
frequency location(index)

% Just to visualize the approximate center frequency index in [Kx Ky Kz]
figure (2)
isosurface(Kx,Ky,Kz,abs(ave),0.75)
axis([-20 20 -20 20 -20 20]), grid on, drawnow
```

```
title('Average signal in frequency domain with isovalue 0.75')
xlabel('Kx')
ylabel('Ky')
zlabel('Kz')

%%
tau = 0.4; % Filter width
k0 = [Kx(x_ind, y_ind, z_ind) Ky(x_ind, y_ind, z_ind) Kz(x_ind, y_ind, z_ind)]; % Frequency
of interest-center frequency
filter = exp(-tau * ((Kx-k0(1)).^2 + (Ky - k0(2)).^2 + (Kz - k0(3)).^2)); % Define the filter
filter = ifftshift(filter);                    % unshift the order of the filter

figure(3)
title('20 Marbles positions in spatial domain with isovalue 0.75 ')
xlabel('x')
ylabel('y')
zlabel('z')
marble_index = zeros(20,3);        % 20 rows for 20 measurement and 3 columns for x y z
coordinates
for i = 1:20
  un(:,:,:) = reshape(Undata(i,:),n,n,n);
  unt = fftn(un);
  unft = filter.*unt;                    % Apply filter to the signal in frequency domain
  unt = ifftn(unft);                     % Transfer the signal back to spatial domain
  unt = abs(unt)./max(unt(:));
  [Max, Ind] = max(unt(:));
  [max_x, max_y, max_z] = ind2sub(size(unt),Ind);
  marble_index(i,:) = [X(max_x, max_y, max_z) Y(max_x, max_y, max_z) Z(max_x, max_y,
max_z)];
  isosurface(X,Y,Z,abs(unt),0.75)
  axis([-20 20 -20 20 -20 20]), grid on, drawnow
end
figure(4)
plot3(marble_index(:,1), marble_index(:,2), marble_index(:,3),'k-');
axis([-20 20 -20 20 -20 20]), grid on
hold on
plot3(marble_index(20,1), marble_index(20,2), marble_index(20,3), 'ro')
title('Marble trajectory with the 20th marble in spatial domain')
xlabel('x')
ylabel('y')
zlabel('z')
```