An Ultrasound Problem

Chenghao Chen
Amath 482

Abstract

In homework 1, we want to use the ultrasound to tract the marble in the dog's body. In order to solve the problem, we are required to use the Fourier Transform to average the data in frequency domain and apply Gaussian filter to find the frequency in spatial domain. Then, we can track marble's position over time, and identify the final location. In this way, we can save poor dog's life.

Introduction

In the problem, the dog Fluffy swallowed a marble accidently. In order to save its life, we need to use ultrasound to locate the marble and take it out as soon as possible. However, the data, which contains 20 rows of data for 20 different measurements that were taken in time, is too noisy because the dog keeps moving. Therefore, we have to denoise the signal before using it. First, we can isolate the frequency of the marble, which we could find the center frequency. Then we can denoise the data by filtering around the center frequency. Finally, we can convert the data back to spatial domain and tract the marble's position.

Theoretical Background

Fourier Transform

The key idea of the Fourier transform is to represent functions and their derivatives as sums of cosines and sines. The Fourier Transform is an integral transform defined over the entire line x ∈ [−∞, ∞]. The Fourier transform and its inverse are defined as

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

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

Fast Fourier Transform

The algorithm associated with the FFT does three major things: it shifts the data so that x ∈ [0, L] → [−L, 0] and x ∈ [−L, 0] → [0, L], additionally it multiplies every other mode by −1, and it assumes you are working on a 2π periodic domain. [1]

Gaussian filter

$$F(k) = \exp\left(-\tau(k - k0)^2\right)$$

where τ measures the bandwidth of the filter, and k is the wavenumber. The generic filter function F(k) in this case acts as a low-pass filter since it eliminates high-frequency components in the system of interest. 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 or wavenumber. [1]

Averaging

white-noise can be modeled adding a normally distributed random variable with zero mean and unit variance to each Fourier component of the spectrum. The key here: with zero mean. Thus over many signals, the white-noise should, on average, add up to zero. A very simple concept, yet tremendously powerful in practice for signal processing systems where there is continuous detection of a signal. [1]

Algorithm Implementation
The first step is to model the data in frequency domain and time domain. The spatial and spectral resolution is given by the problem. Half of the computational domain is 15 and the number of Fourier modes is 64. Then, we discretize the time domain into 21 points since we want 20 identical interval and we only keep first 20 points. Then, we rescaled the frequency by (2pi/2L) because FFT assumes working on a 2pi periodic domain. Also, we set the frequency from 0 to (n/2 -1) and (-n/2 to -1) since FFT shifts the domain. Finally, we create a grid in spatial domain and frequency domain using meshgrid. As shown in figure(1), we have a data picture with white noise.

The second step is to average the spectrum. First, we transform the data from the spatial domain to frequency domain by using fftn. By using a for loop, we will add all new transform together. Out of the for loop, we divide the sum by 20 and get the average. In order to plot the data, we need to use fftshift to shift the transformed function back to the mathematically correct position. Now we know that this will be our marble because the white noise should average themselves out. As shown in figure (2), we have a clear data picture without the white noise.

The third step is to find the center frequency. Since the center frequency will have highest value, we use max to locate the maximum value and find the center frequency.

```
[M,I] = max(Uave(:))
```

M will be the maximum value and I will be index, which is the center frequency. Then, we use ind2sub to convert linear indices to subscripts. Finally, we can find the center frequency K_center.

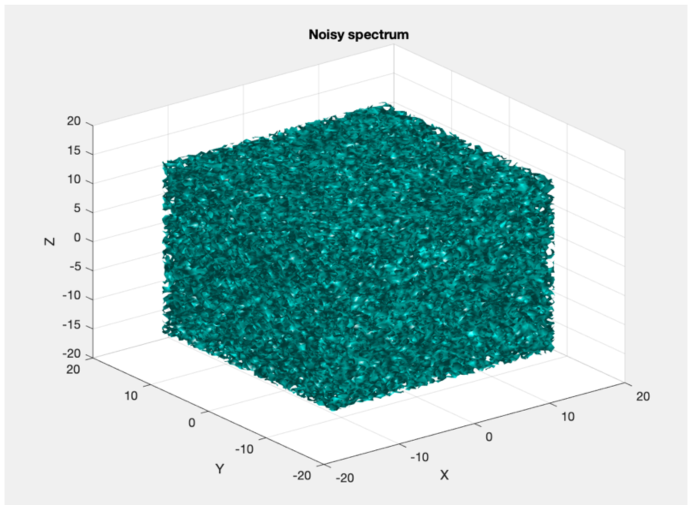
The fourth step is to filter the data around the center frequencies and find the marble. A 3-D Gaussian filter is

$$F(k) = e^{-[(kx-k_a)^2+(ky-k_b)^2+(kz-k_c)^2]}$$

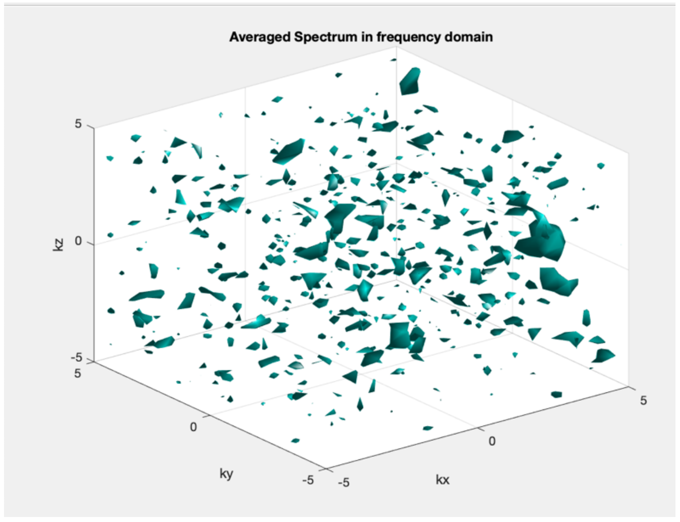
Where (ka,kb,kc) is the location of the marble and the bandwidth is 1. This allows us to focus only on the marble frequency and ignore others. First, we transform the noisy data by using fftn. Then, we filter the data by multiply the filter. After we filter the data, we transform the filtered data back to spatial domain. Last but not least, we find the coordinate of the marble and store the position in the data. The coordinate of the marble is the indices of the maximum value, and we use the same method in the third step to find the coordinate. As shown in figure(3), we find out the position of the marble.

Now we have the position of the marble and we can target the final position to help Fluffy.

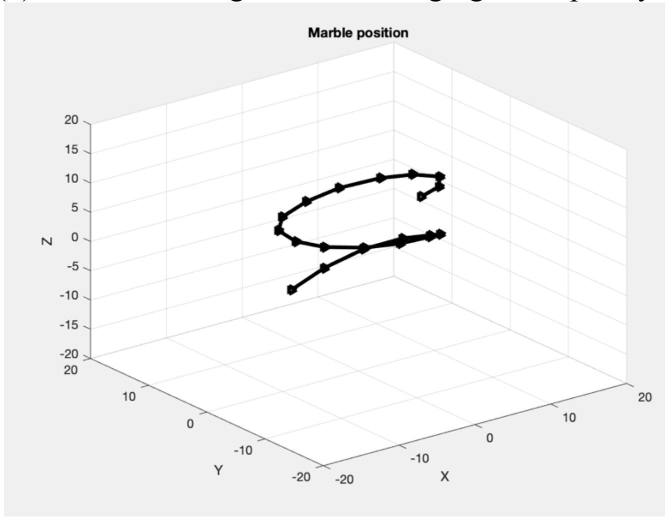
Computational Result
Graphics below are the process of the data analysis from original data to final marble trajectory.



Figure(1) : Ultrasound signal with a huge amount of white noise in time domain



Figure(2) : Ultrasound signal after averaging in frequency domain



Figure(3) : marble position in time domain

Based on the marble position in figure(3), we can find out the last position of the marble at 20th measurement.
x = -5.6250   y = 4.2188   z = -6.0938
Also, the center frequency is
kx = 1.8850   ky = -1.0472   kz=0

Summary and conclusion

We successfully save Fluffy. In this homework, I learn to move the signal to frequency domain by using the Fourier Transform. Also, I can reduce the white noise and find the center frequency by averaging all the measurements. Moreover, a Gaussian filter is an effective way to filter the signal around the center frequency. With this assignment, I realize how powerful the Fourier Transform is. I also learn how to model a signal from 2D to 3D and some useful MATLAB command. Overall, it is a useful and fun assignment.

Citation

[1] J. N. Kutz, *Data-driven modeling & scientific computation: methods for complex systems & big data*. Oxford: Oxford University Press, 2013.
[2] “Makers of MATLAB and Simulink,” *MathWorks*. [Online]. Available: https://www.mathworks.com/. [Accessed: 25-Jan-2020].

Appendix A : MATLAB function

fftn(N) : This MATLAB function returns the multidimensional Fourier transform of an N-D array using a fast Fourier transform algorithm. [2]

fftshift(N) : This MATLAB function rearranges a Fourier transform X by shifting the zero-frequency component to the center of the array. [2]

meshgrid() : This MATLAB function returns 2-D grid coordinates based on the coordinates contained in vectors x and y. [2]

reshape() : This MATLAB function reshapes A using the size vector, sz, to define size(B). [2]

isosurface() : This MATLAB function computes **isosurface** data from the volume data V at the **isosurface** value specified in isovalue. [2]

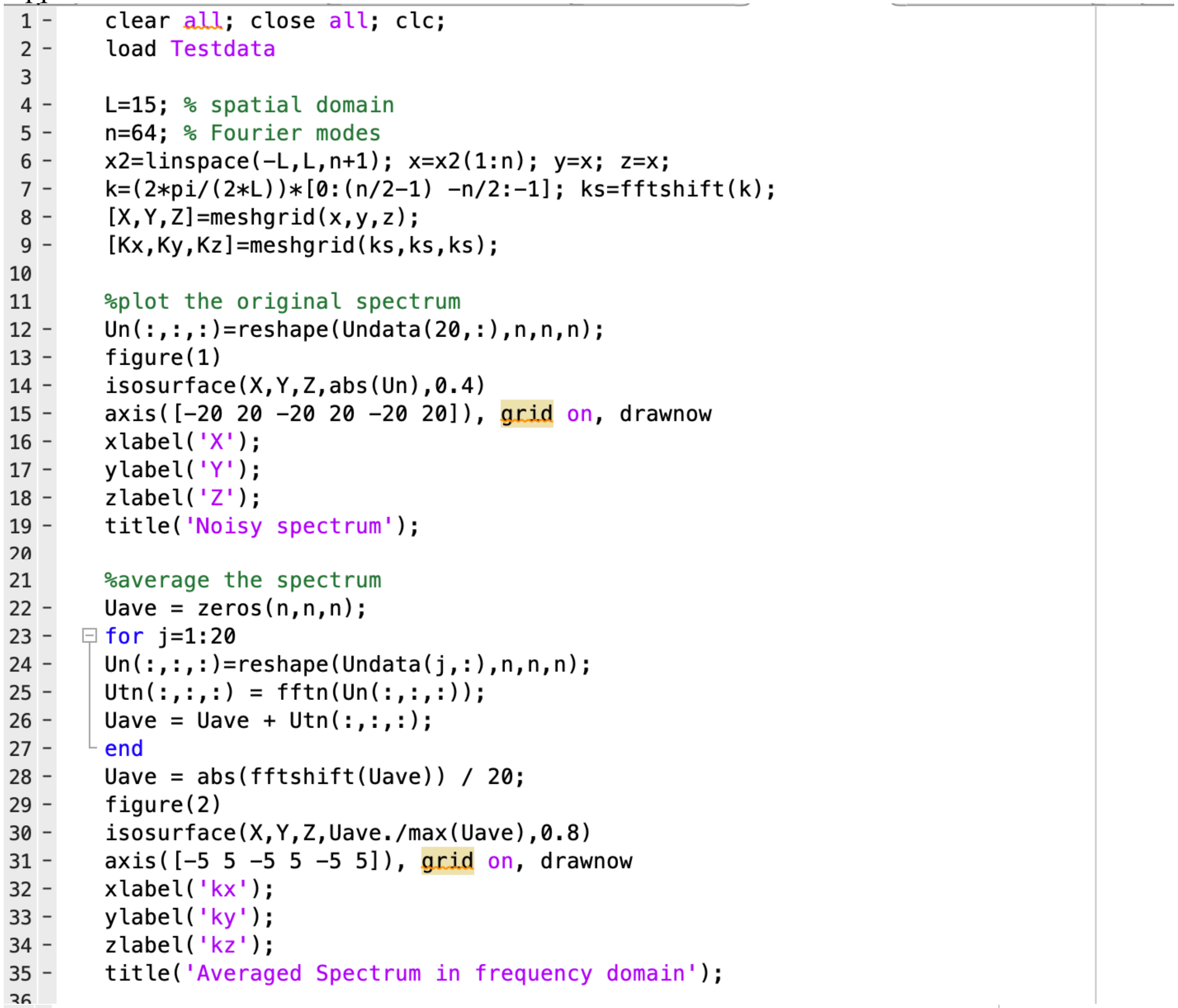
ifftn() : This MATLAB function returns the multidimensional discrete inverse Fourier transform of an N-D array using a fast Fourier transform algorithm. [2]

ind2sub() : This MATLAB function returns the arrays row and col containing the equivalent row and column subscripts corresponding to the linear indices ind for a matrix of size sz. [2]

max(N) : This MATLAB function returns the maximum elements of an arra. [2]

plot3() : This MATLAB function plots coordinates in 3-D space. [2]

Appendix B: MATLAB code

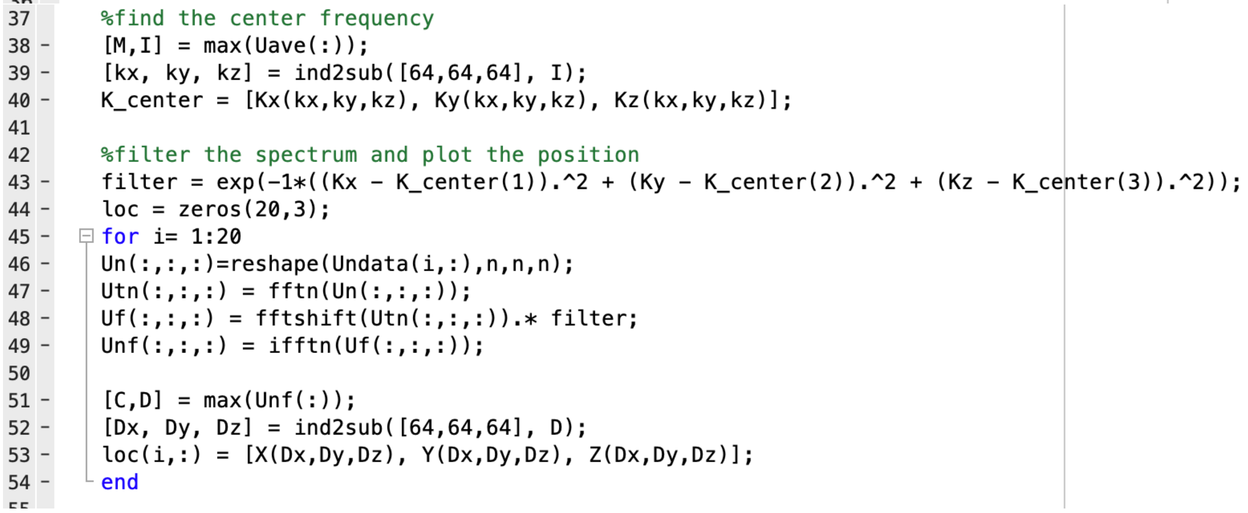


```matlab
clear all; close all; clc;
load Testdata

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);

%plot the original spectrum
Un(:,:,:)=reshape(Undata(20,:),n,n,n);
figure(1)
isosurface(X,Y,Z,abs(Un),0.4)
axis([-20 20 -20 20 -20 20]), grid on, drawnow
xlabel('X');
ylabel('Y');
zlabel('Z');
title('Noisy spectrum');

%average the spectrum
Uave = zeros(n,n,n);
for j=1:20
Un(:,:,:)=reshape(Undata(j,:),n,n,n);
Utn(:,:,:) = fftn(Un(:,:,:));
Uave = Uave + Utn(:,:,:);
end
Uave = abs(fftshift(Uave)) / 20;
figure(2)
isosurface(X,Y,Z,Uave./max(Uave),0.8)
axis([-5 5 -5 5 -5 5]), grid on, drawnow
xlabel('kx');
ylabel('ky');
zlabel('kz');
title('Averaged Spectrum in frequency domain');
```



```matlab
%find the center frequency
[M,I] = max(Uave(:));
[kx, ky, kz] = ind2sub([64,64,64], I);
K_center = [Kx(kx,ky,kz), Ky(kx,ky,kz), Kz(kx,ky,kz)];

%filter the spectrum and plot the position
filter = exp(-1*((Kx - K_center(1)).^2 + (Ky - K_center(2)).^2 + (Kz - K_center(3)).^2));
loc = zeros(20,3);
for i= 1:20
Un(:,:,:)=reshape(Undata(i,:),n,n,n);
Utn(:,:,:) = fftn(Un(:,:,:));
Uf(:,:,:) = fftshift(Utn(:,:,:)).* filter;
Unf(:,:,:) = ifftn(Uf(:,:,:));

[C,D] = max(Unf(:));
[Dx, Dy, Dz] = ind2sub([64,64,64], D);
loc(i,:) = [X(Dx,Dy,Dz), Y(Dx,Dy,Dz), Z(Dx,Dy,Dz)];
end
```

```matlab

    figure(3)
    plot3(loc(:,1), loc(:,2), loc(:,3), 'k->', 'Linewidth', 3);
    axis([-20 20 -20 20 -20 20]);
    xlabel('X');
    ylabel('Y');
    zlabel('Z');
    title('Marble position');
    grid on

    loc_20 = loc(20,:);

```