An Ultrasound Problem

Khoa Vo

Abstract: In the real world, the data given to us is not always nicely represented, but highly noisy. In order to use the data for different purposes, our job is to eliminate the noise in the data first. In this task, we went through a process of determine the spectral signature and extract location of the marble at 20 different moments inside a dog's intestines from high noisy data by using Fast Fourier Transfer, Averaging, and Gaussian Filter.

1. Introduction and Overview

My dog fluffy swallowed a marble, which has worked its way into the intestines. The data concerning the spatial variations inside the intestines is obtained by using Ultrasound. Unfortunately, the original data is highly noisy due to fluffy keeps moving and the internal fluid movement. The data has 20 rows of data for 20 different measurements taken in time.

In order to save fluffy, we have to denoise the data and determine location of the marble inside the intestines. An intense acoustic wave can be used to breakup the marble at the 20th measurement.

2. Theoretical Background

We will need 3 key concepts to denoise the data and determine the path of the marble inside the intestines: Fourier Analysis, Averaging, and Gaussian Filtering.

2.1. Fourier Analysis

Fast Fourier Transform (FFT) is an algorithm that is widely used for applications in engineering, data science, music, and mathematics. FFT algorithm converts the signal in the original domain (time or space) into a representation in frequency domain. FFT algorithms are known to have time complexity of $O(N \log N)$. It finds the transform on the interval $x \in [-L, L]$ and discretizes the range of x into 2^n points. FFT is defined by the formula:

$$X_k = \sum_{n=0}^{N-1} x_n e^{-i2\pi kn/N} \tag{1}$$

Its inverse (IFFT) is defined by the formula:

$$x_k = 1/N \sum_{n=0}^{N-1} X_k e^{-i2\pi kn/N} \tag{2}$$

2.2. Averaging The Spectrum

We need to use the fact that noise is white and it is normally distributed random variable. The key idea is that white-noise has mean zero because it is normally distributed. Therefore, the white-noise over many signals should add up to zero on average. We can use Averaging to ultimately extract the spectral signature of the data.

2.3. Gaussian Filtering

In real world, the signal can be buried inside noise field due the geographical objects or the interaction with other signals. Gaussian Filtering is a method that allow us to extract the signal at specific frequencies, which means if we have the spectral signature, we can filter white-noise around it. Gaussian Filter:

$$\mathcal{F} = exp(-\tau(k - k_0)^2) \tag{3}$$

where τ is the bandwidth of the filter and k_0 is the spectral signature of the signal. In higher dimension, the Gaussian filtering is used by adding k_i variables.

3. Algorithm Implementation and Development

First, We will set up the x, y, z axis on the domain L = 15. Then, since FFT shifts the data so we will need to define the frequency domain k from 0 to $n/2-1$ and $-n/2$ to -1, in which n = 64 and then define ks by shifting k. In MATLAB, we will use the command "fftn" to compute the Fast Fourier Transform for 3 dimensions, and it assumes that we are working on the 2π periodic domain. Therefore, we need to rescale the frequency domain by the factor of $2\pi/2L$. Finally, we need to transform the domain of 3 vectors x, y, and z into arrays used to evaluate the function in spatial domain, [X, Y, Z]. Also, I need to transform ks into arrays used to evaluate the function in frequency domain, [Kx, Ky, Kz].

```
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);
```

3.1. Determine The Frequency Signature by Averaging The Spectrum

The original data is 20 rows representing the measurement at 20 different moments in time. We want to reshape each row into the size of 64*64*64 along x, y, z axis, use Fast Fourier Transform to converts them into a representation in frequency domain, and sum them up. We can get the average spectrum by dividing the sum by the number of rows, 20. Since, the 'fftn' shift the data, we need to use 'fftshift' to shift the transformed function back to its mathematically correct position. The original data are complex numbers, but for this task, we only care about the real part of the data, hence, we use 'abs' to get the result. After doing those steps, the data now is noise-reduced. The frequency we have is reflecting the marble. To get the spectral signature, we need to find the indices, where has the maximum |fftn| value in frequency domain and use those indices to locate the spectral signature of the data.

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

utn_ave = abs(fftshift(utn./20));% averaging
index = find(utn_ave == max(utn_ave(:)));%get indices
x_ss = Kx(index);
y_ss = Ky(index);
z_ss = Kz(index);
```

3.2. Filter Data

We need to construct the Gaussian Filter to get the desired signal field in relation to the center frequency (spectral signature) we found on the previous section. In this task, we will use bandwidth of 0.5 to capture the main signal.

It is significant that we shift the filter, because vector [Kx, Ky, Kz] is constructed by transforming vector 3 ks, which is shifted from vector k.

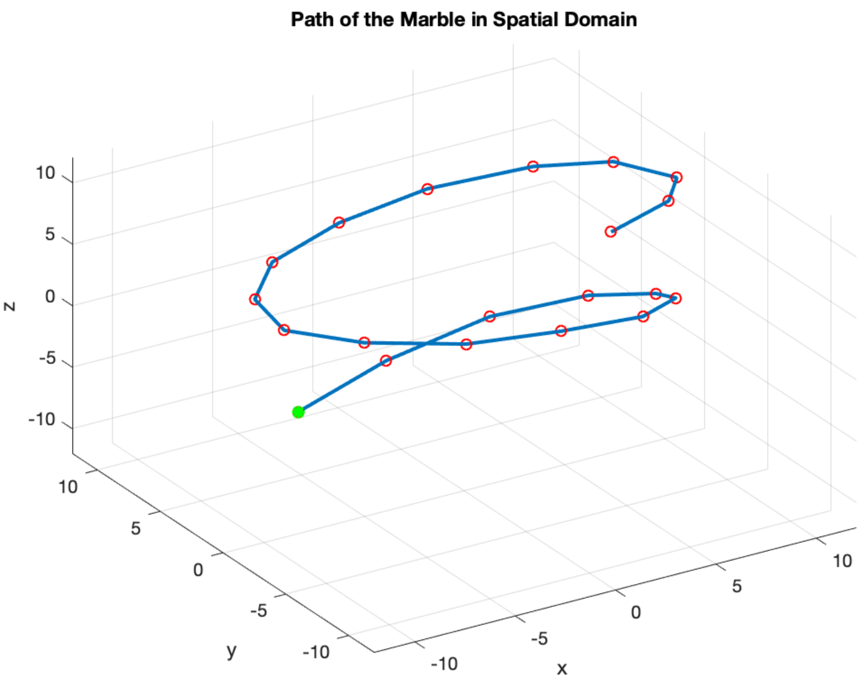
In order to filter the data, we need to go through 20 rows and reshape them into the size of 64*64*64 along the x, y, z axis. Then, transform each into a representation in frequency domain and multiply them by the Gaussian Filter. Following that, I need to convert them back to the spatial domain by using Inverse Fast Fourier Transform. The data we have now is the correct location of the marble in x, y, z coordinates.

```
%filter
tau = 0.5;
coord = zeros(20, 3);
filter = exp(-1*tau*((Kx - x_ss).^2 + (Ky - y_ss).^2 +(Kz - z_ss).^2));
filter = fftshift(filter);%shift the filter
for i=1:20
```

```
    Un(:,:,:)=reshape(Undata(i,:),n,n,n);
    utn = fftn(Un);
    unft=filter.*utn;
    unf = ifftn(unft);
    index = find(unf == max(unf(:)));
    x_sd = X(index);
    y_sd = Y(index);
    z_sd = Z(index);
    coord(i,:) = [x_sd y_sd z_sd];
end
```

4. Computational Results

The Spectral Signature of the marble inside the intestine is [1.8850 -1.0472 0]



482/homework1matlab111.png

The intense wave should be focused at [-5.6250 4.2188 -6.0938] (20th measurement of and the green point in the Figure above) to break up the marble to save the dog.

5. Summary and Conclusions

Fast Fourier Transform, Averaging, and Gaussian Filter are very powerful tools to eliminate the noise in our data. We went through a process of determining the center frequency, using Gaussian filter to extract the path of marble, and coming up with an answer where the vet should focus the intense wave to save the dog.

Appendix A.

fftn() : Fast Fourier Transform in n-dimension
ifftn() : Inverse Fast Fourier Transform in n-dimension
fftshift(): Rearrange a Fourier Transform by shifting the zero-frequency component to the center
reshape(): Reshape the data into n dimensions

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);

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

%averaging
utn_ave = abs(fftshift(utn./20));% averaging
index = find(utn_ave == max(utn_ave(:)));%get indices
x_ss = Kx(index);
y_ss = Ky(index);
z_ss = Kz(index);

% filter
tau = 0.5;
coord = zeros(20, 3);
filter = exp(-1*tau*((Kx - x_ss).^2 + (Ky - y_ss).^2 +(Kz - z_ss).^2));
filter = fftshift(filter);
for i=1:20
    Un(:,:,:)=reshape(Undata(i,:),n,n,n);
    utn = fftn(Un);
    unft=filter.*utn;
    unf = ifftn(unft);
    index = find(unf == max(unf(:)));
    xc = X(index);
    yc = Y(index);
    zc = Z(index);
    coord(i,:) = [xc yc zc];
end
plot3(coord(:,1), coord(:,2), coord(:,3), 'Linewidth', [2]), grid on; hold on;
plot3(coord(:,1), coord(:,2), coord(:,3), 'ro'); hold on;
plot3(coord(20,1), coord(20,2), coord(20,3), 'g*', 'Linewidth', [2])
xlim([-12, 12]);
ylim([-12, 12]);
zlim([-12, 12]);
xlabel('x');
ylabel('y');
zlabel('z');
```