Homework1: An ultrasound problem

Minghu Zhou

January 20th, 2020

Abstract

My dog fluffy swallowed a marble and I want to locate the marble in its intestines. In this report I initially have a dataset of 20 different measurements taken in time by ultrasound. I average and filter the noisy signal to get useful information on these data. Using Fourier transform, we can convert the signal to the frequency domain. Then we can know the center frequency by averaging the spectrums and apply a Gaussian filter to denoise the data. After an inverse Fourier Transform, we are able to have the marble's locations of 20 measurements and thus we can plot its trajectory.

1. Introduction and Overview

The marble is now in the dog's intestines where there are fluids. This means the fluids will add noise to the data in spatial domain. However, the noisy dataset can still have a frequency signature (center frequency) in its frequency domain. Fast Fourier transform (FFT) helps us transform the dataset to its spectrum, and since the noise is caused by the random movement of the fluids, we average the spectrums to find the frequency signature. A Gaussian filter centered around the frequency signature is used to remove the noise. With Inverse Fast Fourier transform (IFFT), we convert the denoised frequency data back to spatial domain and thus can plot its locations and trajectory clearly. With the information of the marble's trajectory, an intense acoustic wave can be used at the final position of the marble to break it and the dog can be saved.

In this report, I will first introduce FFT and IFFT, Spectrum averaging, and Gaussian Filtering. Next I will talk about the algorithms implementation in MATLAB and show the computational results. I will also include MATLAB functions used and code in the end.

2. Theoretical Background

2.1 Fourier Transform and FFT

We use Fourier Transform to decompose a function, especially a time signal into frequency representation. The Fourier Transform of function f(x) gives a frequency function F(k):

$$F(k) = \frac{1}{2\pi}\int_{-\infty}^{\infty} e^{-ikx} f(x)dx \qquad (2.1)$$

Here k denotes the frequency and F(k) represents the strength of the corresponding frequency.

The inverse Fourier Transform converts a function in frequency domain back to its time domain:

$$f(x) = \frac{1}{2\pi}\int_{-\infty}^{\infty} e^{ikx}F(k)dk \qquad (2.2)$$

The Fast Fourier Transform (FFT) is a numerical method of integration. It divides the signal into two pieces recursively, so it requires the signal to have 2^n points. Meanwhile, FFT assumes 2π periodic signal, so we have to scale our data to fit it.

2.2 Spectrum averaging

Since the noises are all random, the sum of the white noise of multiple measurements will be close to zero. By averaging the signals in the frequency domain, most of the noise can be cleared out. Thus, the center signal which have the strongest strength will be distinct so that we are able to locate it.

2.3 Gaussian filtering

To remove the noise of a signal, we can use a filter to attenuate the signal not near the interested frequency. Specifically, we can multiply the signal in frequency domain by a filter function which has a value of near 0 for frequencies that are not interested. In this way, the interested frequency will be extinguishable.

Gaussian function is a common filter function:

$$F(k) = e^{-\tau(k-k_0)^2} \qquad (2.3)$$

The Gaussian function is centered at k_0 and has a value of 1. The farther away from k_0 , the closer the value is to 0. τ represents the decreasing rate.

3. Algorithms Implementation and Development

3.1 Load data and set up the domain

We first load the ultrasound data from Testdata.mat, ***Undata***, which is a 20 * n^3 matrix.

Then, every axis has n points spaced equally from $[-L, L)$ in the spatial domain. We rescale frequencies by multiplying modes in $[-n/2, n/2)$ by $2\pi / L$ since the FFT assumes 2*pi periodic signals, so every axis has n points spaced equally from $[-\pi, \pi)$ in the frequency domain. We also swap the first half of the frequency domain and the second half as FFT function in MATLAB shifts the input data in that way.

Since there are 3 axes in this problem, to get all positions in spatial domain we have to make a 3D grid of points in each direction, each has a size of $n * n * n$ and so does it in the frequency domain. We also reshape each row of the loaded matrix to a $n * n * n$ grid to corresponds with those in spatial and frequency domain.

3.2 Average spectrums and Find Center Frequency

In this part, we average the spectrums to remove the noise and in this way we find the center frequency.

First, we apply FFT on 20 measurements and take average of them. The result averaged spectrum will have the strength of frequencies caused by random noise decreased, leaving the frequencies of the marble.

In order to find the center frequency, we find the maximum absolute value and the corresponding index of the averaged spectrum. Apply that index to the 3D grid of each direction in the frequency domain, we can get the center frequency.

3.3 Filter Data to get the locations of marble

Now we already have the center frequency in the frequency domain, to get the location of the marble in each time point we need to filter the noise of every measurement.

We first create a 3D Gaussian filter centered at the center frequency in the Fourier space. Notice that we need to ***fftshift*** the filter to match the order of the Fourier-transformed data because FFT function in MATLAB swaps the first half and the second.

Then, we repeat the following process for each measurement:

1. Multiply the transformed data by the Gaussian function to remove the random noise.

2. Use ***ifft*** in MATLAB to convert the filtered data back to time domain.

3. Find the index corresponding to the maximum intensity among the filtered data in the spatial domain.

4. Apply that index to the 3D grid in each direction in the frequency domain, we can get and record the location of the marble at each time point.

After repeating the above process for 20 measurements, we now have 20 locations of the marble from beginning to the end. We can 3D plot the trajectory of the marble with these locations using ***plot3*** function in MATLAB.

4. Computational Results

1. The center frequency $[k_x, k_y, k_z]$ computed by algorithm in 3.2 is $[1.885, -1.047, 0]$

2. The trajectory of the marble over time computed by algorithm in 3.3 is plotted in ***figure 1***.

3. The position of the marble at the 20th measurement is $[-5.625, 4.219, -6.094]$.

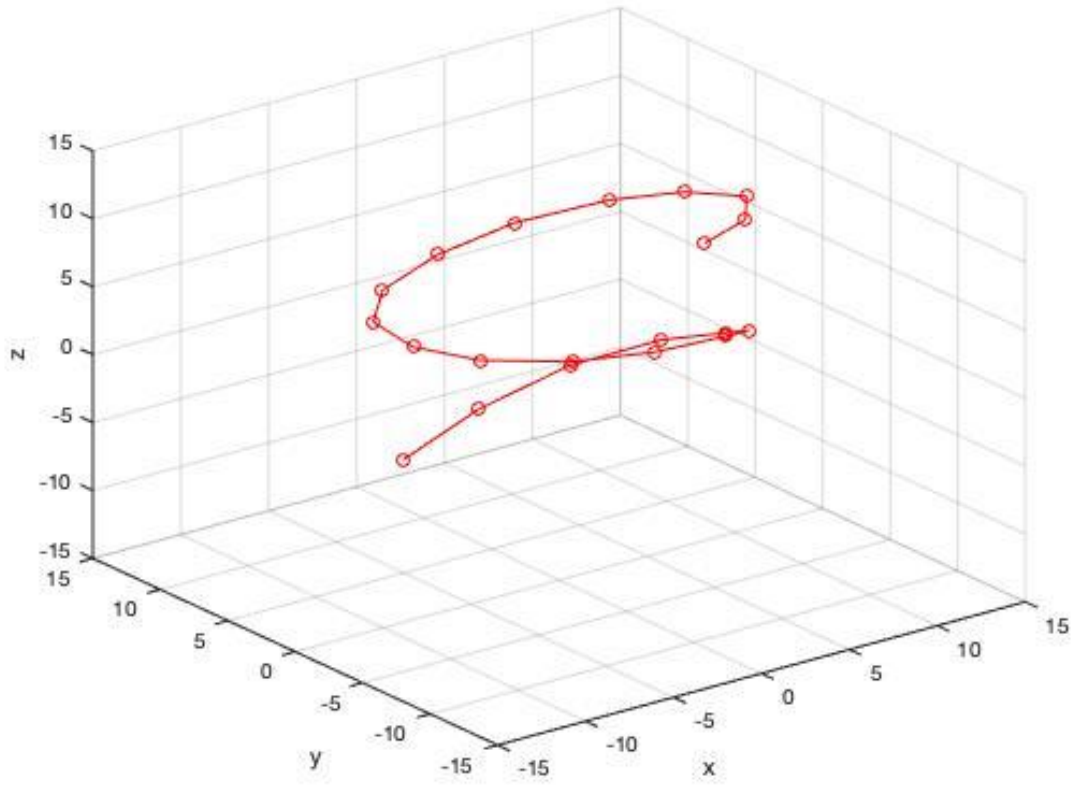

Figure 1. trajectory of the marble

5. Summary and Conclusions

In conclusion, we are able to determine the position of the marble using Fast Fourier Transform, spectrum averaging and Gaussian filter and we successfully denoise the data we have.

An intense acoustic wave should be focused at (−5.625, 4.219, −6.094) in order to break up the marble at the 20th data measurement.

Appendix A.
MATLAB functions and brief implementation explanation

- $linspace(x1, x2, n)$: Return n points. The spacing between the points is (x2-x1)/(n-1).
- $fftshift(X)$: Rearranges a Fourier transform X by shifting the zero-frequency component to the center of the array.

- $meshgrid(x, y, z)$: 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).
- $reshape(A, sz1, \ldots, szN)$: Reshapes A into a sz1-by-...-by-szN array where sz1,...,szN indicates the size of each dimension.
- $isosurface(X, Y, Z, V, isovalue)$: Computes isosurface data from the volume data V at the isosurface value specified in isovalue.
- $fftn(X)$: Returns the multidimensional Fourier transform of an N-D array using a fast Fourier transform algorithm.
- $ifftn(Y)$: Returns the multidimensional discrete inverse Fourier transform of an N-D array using a fast Fourier transform algorithm.
- $plot3(X1, Y1, Z1, \ldots)$: Plots one or more lines in three-dimensional space through the points whose coordinates are the elements of X1, Y1, and Z1.

Appendix B.
MATLAB codes

```
clear; close all; clc;

%% load data and set up the domain
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);
% for j=1:20
%    Un(:,:,:)=reshape(Undata(j,:),n,n,n);
%    close all, isosurface(X,Y,Z,abs(Un),0.4)
%    axis([-20 20 -20 20 -20 20]), grid on, drawnow
%    pause(1)
% end

%% Average spectrums and Find Center Frequency
avgUnt = zeros(64,64,64);
for j=1:20
   Un(:,:,:)=reshape(Undata(j,:),n,n,n);
   Unt = fftn(Un);
   avgUnt(:,:,:) = avgUnt + Unt;
end
avgUnt = abs(fftshift(avgUnt))/20;
[maximum,index] = max(avgUnt(:));
xc = Kx(index)
yc = Ky(index)
zc = Kz(index)

%% Filter Data to get the locations of marble
places = zeros(20,3);
```

```
filter = exp(-1.0*fftshift(Kx-xc).^2).* ...
    exp(-1.0*fftshift(Ky-yc).^2) .* ...
    exp(-1.0*fftshift(Kz-zc).^2);

for j = 1:20
    Un(:,:,:)=reshape(Undata(j,:),n,n,n);
    Unt = fftn(Un);
    unft = filter.* Unt;
    unf = ifftn(unft);
    unf = abs(unf);
    [maximum,index] = max(unf(:));
    places(j,:) = [X(index) Y(index) Z(index)];
    %isosurface(X,Y,Z,unf/maximum,0.75)
    %axis([-20 20 -20 20 -20 20]), grid on, drawnow
end
figure()
plot3(places(:,1), places(:,2), places(:,3), 'ro-')
axis([-L L -L L -L L]), xlabel('x'); ylabel('y');zlabel('z');
grid on, drawnow

fplace = places(20,:)
```