Homework 1: An ultrasound problem

Yilin Li

January 24 2020

Abstract

Determine the frequency signature from highly noisy data through averaging of the spectrum. The noisy data were taken in time in a small area of the intestines where a marble is suspected to be. Use Gaussian filter to filter the data around the frequency signature in order to denoise the data and determine the path of the marble. In the end, figure out where an intense acoustic wave should be focused to breakup the marble at the 20th noisy data measurement. All of above were solved by using MATLAB software

Sec. I. Introduction and Overview

A dog swallowed a marble, and the vet thought it had been into the intestines. When the vet used ultrasound to obtain data in order to find where the marble was by concerning the spatial variations in the intestines, the dog kept moving and the internal fluid movement through the intestines generated highly noisy data.

In this project, the Fast Fourier Transform (FFT) algorithm, averaging the spectrum, determining the frequency signature, and the Gaussian Filter are implemented to find the marble and save the dog.

The Theoretical Background section introduces the theorems which were used in the project.

The Algorithm Implementation and Development section lists the algorithms that were implemented in this project and how they were developed.

The Computational Results section showed the plots and analyzed the computational results of MATLAB.

The Summary and Conclusion section summarized the project and announced the fate of the poor dog.

The MATLAB functions and implementations used in this project were attached in Appendix A. The MATLAB codes were attached in Appendix B.

Sec. II. Theoretical Background

2.1 The Fourier Transform

The Fourier Transform was initially brought up as an analytic tool for thermal process. The big idea of the Fourier Transform were taking a signal and breaking it down into different frequencies or taking a function and breaking it down into sines and cosine of different frequencies. It is defined over the entire line, x $\in [-\infty, \infty]$. However, we usually do not use it as the interval because it goes forever. The domain that was used in this project is x $\in [-L, L]$ because the continuous eigenfunction expansion becomes a discrete sum of eigenfunctions and associated eigenvalues on a finite domain. The Fourier Transform is defined as:

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

In time of frequencies, the Fourier Transform does not provide the information of the position because a narrow signal in the time domain has a large spread in the frequency domain and vice-versa.

2.2 The Inverse Fourier Transform

The inverse Fourier transform is extremely similar to the original Fourier transform. It differs only in the application of a flip operator. The inverse Fourier transform is defined as:

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

2.3 The Fast Fourier Transform (FFT)

The Fast Fourier transform routine was developed specifically to perform the forward and backward Fourier transforms. In the mid 1960s, Cooley and Tukey developed the FFT algorithm. The key features of the FFT routine are as follows: 1. It has a low operation count: O(N log N). 2. It finds the transform on an interval x $\in [-L, L]$. 3. The key to lowering the operation count to O(N log N) is in discretizing the range x $\in [-L, L]$ into 2n points. 4. The FFT has excellent accuracy properties, typically well beyond that of standard discretization schemes.

2.4 The Averaging of the Spectrum

Averaging the frequency spectrum is an effective method to suppress white noise (a random signal equally distributed over all frequencies with zero mean and finite standard deviation). In this project, the averaging of the spectrum was used to filter signals in which the frequency of the signal is an unknown constant in order to find the frequency signature.

2.5 The Gaussian Filter

Gaussian Filter is a useful tool to filter noisy data. A Gaussian Filter centered on the frequency of interest with its width τ. The function is defined as:

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

It is important to have the right frequency of interest, k_0. In practice we may need to try different τ to see which one works better.

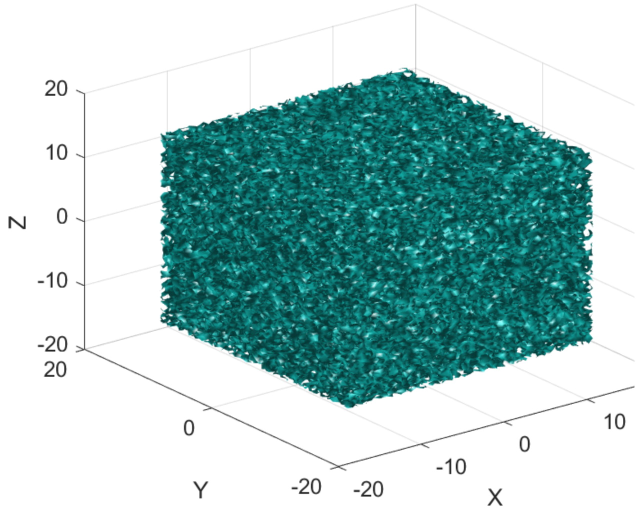
Sec. III. Algorithm Implementation and Development

- First of all, load Testdata;
- Define the spatial domain L = 15 and the Fourier modes n = 64;
- Define the domain discretization and grid vector;
  - The grid vectors for the frequency domain must be multiplied by $\frac{2\pi}{L}$ because the FFT algorithm assumes 2π period signals.
  - The grid vectors k must be shifted by $fftshift$ function because the FFT algorithm shifts x $\in [-L, 0] \to [0, L]$ and x $\in [0, L] \to [-L, 0]$.
  - $fftshift$ is only used for plot, it does not matter when do calculation
- Return 3-D grid coordinates based on the coordinates in x,y,z using $meshgrid$
- Calculate the average frequencies for the 20 different measurements;
  - Reshape $Undata$ into a 64-by-64-by-64 array
  - Fourier transform each data signal using $fftn$ function and add together
- Find the frequency signature (center frequency) in Utave
  - Normalize $Utave$
  - Use $forloop$ to find the largest element in the matrix $Utave$ record the location of the element (the center frequency) to set the Gaussian filter later;
  - Take the absolute value of $Utave$ because the FFT multiplies every other mode by -1
- Create the $isosurface$ of frequency signature
- Create the Gaussian filter with width $\tau = \frac{1}{2}$ (found after tried sometimes) and center $ks(a), ks(b), ks(c)$ that was found above, the location of the largest element
  - Print out $ks(a), ks(b), ks(c)$ to check which one of the shift of the k vector belongs to Kx, Ky, Kz

- Apply the Gaussian filter to denoise the data by multiplying each signal in Fourier space by the Gaussian filter
  - Take the inverse FFT of each signal
  - Use $fftshift$ to shift the data
  - Normalize the data that was applied the filter using the largest element in the matrix and take the absolute value (same as above, for the same reason)
  - Store the indicies of all center frequencies which is the path of the marble
- Use $plot3$ to plot the path of the marble
- Find out the location of the marble at the $20th$ data measurement

Sec. IV. Computational Results

- Figure 1 shows the isosurface of noisy data in time domain, we cannot observe the frequency signature at all;
- Through averaging of the spectrum, the frequency signature (center frequency) generated by the marble is concentrated at (9,-5,0) (after multiplied by $\frac{L}{2\pi}$);
- Figure 2 shows the the frequency signature of the signal through average of the spectrum
- Figure 3 shows the path of the marble, and the marble moves downward helically
- Figure 4 shows the final location of the marble, and an intense acoustic wave should be focused at (-5.6250,4.2188,-6.0938).



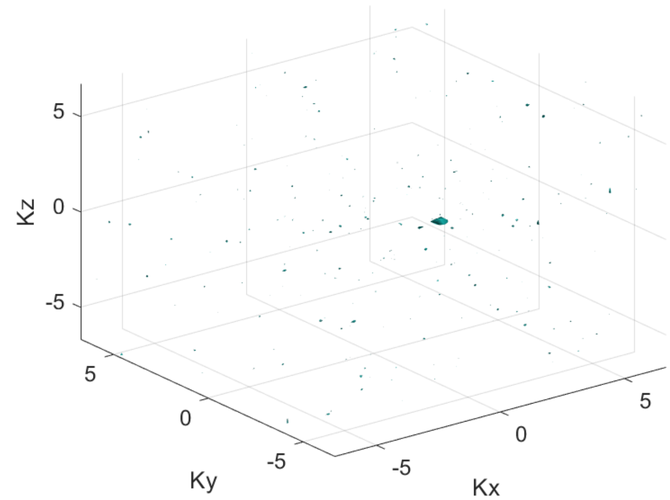
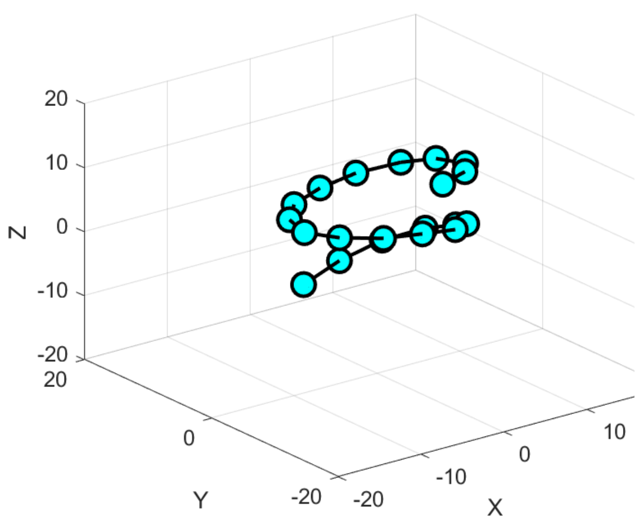


Figure 1: Isosurface of noisy data in time domain

Figure 2: the frequency signature of the signal



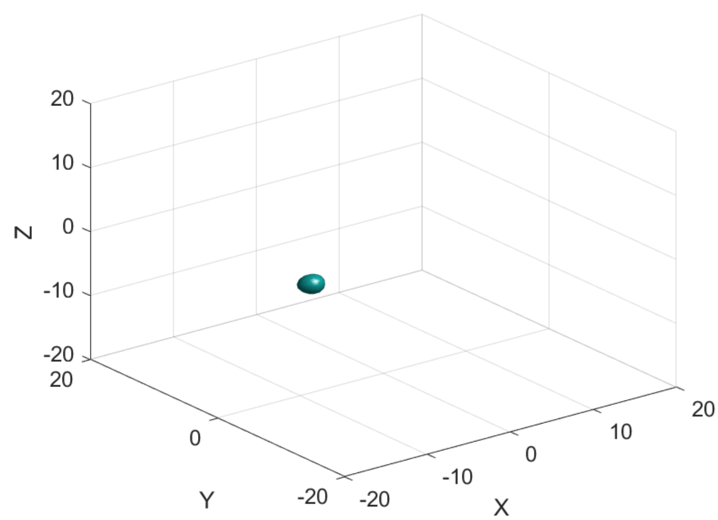


Figure 3: the path of the marble

Figure 4: the final location of the marble

Sec. V. Summary and Conclusions

The averaging of the spectrum and the Gaussian filter was successfully applied to solve the problem. Through averaging of the spectrum, the frequency signature generated by the marble was determined; the data was filtered and denoised; the path of the marble was determined; and the last location of the marble was figured out. The intense acoustic wave could be implemented to save the dog.

Appendix A.

MATLAB functions and implementation

- **linspace(a,b,n)** generates a row vector of n linearly spaced points between a and b.
- **fftshift(X)** rearranges a Fourier transform X by shifting the zero-frequency component to the center of the array.
- **abs(X)** returns the absolute value of each element in array X.
- **[X,Y] = meshgrid(x,y)** returns 2-D grid coordinates based on the coordinates contained in vectors x and y.
- **zeros(n,n,n)** returns an n-by-n-by-n matrix of zeros.
- **B = reshape(A,sz)** reshapes A using the size vector, sz, to define size(B).
- **Y = fftn(X)** returns the multidimensional Fourier transform of an N-D array using a fast Fourier transform algorithm.
- **fv = isosurface(X,Y,Z,V,isovalue)** computes isosurface data from the volume data V at the isosurface value specified in isovalue.
- **X = ifftn(Y)** returns the multidimensional discrete inverse Fourier transform of an N-D array using a fast Fourier transform algorithm.
- **plot3(X,Y,Z)** plots coordinates in 3-D space.
- **grid on** displays the major grid lines for the current axes or chart returned by the gca command.
- **gca** returns the current axes or chart for the current figure, which is typically the last one created or clicked with the mouse.
- **axis(limits)** specifies the limits for the current axes. Specify the limits as vector of four, six, or eight elements.

Appendix B. MATLAB codes

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

```

```
L = 15; % spatial domain
n = 64; % Fourier modes

% Define the domain discretization
x2 = linspace(-L,L,n+1);
% Consider only the first n points (periodicity)
x = x2(1:n);
y = x;
z = x;
% frequency components
k = (2*pi/(2*L))*[0:(n/2-1) -n/2:-1];
% Shift the k vector so the frequencies match up (use for plot)
ks = fftshift(k);

% Return 3-D grid coordinates based on the coordinates in x, y, z
[X,Y,Z] = meshgrid(x,y,z);
[Kx,Ky,Kz] = meshgrid(ks,ks,ks);

% Create variable for average frequency
Utave = zeros(n,n,n);

% Calculate average frequencies for 20 different measurements
for j = 1:20
    % Reshape Undata into a 64-by-64-by-64 array
    Un(:,:,:) = reshape(Undata(j,:),n,n,n);
    Unt(:,:,:) = fftn(Un);
    Utave = Utave + Unt;
end

% ---------------------------------------- Figure 1 ----------------------------------------
%
% Draw the isosurface in time domain
figure(1)
isosurface(X,Y,Z,abs(Un),0.4);
axis([-20 20 -20 20 -20 20]), grid on;
xlabel('X');ylabel('Y');zlabel('Z')
set(gca,'Fontsize',12)
%

Utaves = fftshift(Utave);
absUtaves = abs(Utaves);

% Find the frequency signature (center frequency) in Utave
cfreq = 0;
```

```
for ii = 1:n
    for jj = 1:n
        for kk = 1:n
            if absUtaves(ii,jj,kk) > cfreq
                cfreq = absUtaves(ii,jj,kk);
                a = ii;
                b = jj;
                c = kk;
            end
        end
    end
end

% ------------------------------------------ Figure 2 ------------------------------------------

% Create the isosurface of frequency signature
figure(2)
isosurface(Kx,Ky,Kz,absUtaves/cfreq,0.6);
axis([-abs(ks(1)) abs(ks(1)) -abs(ks(1)) abs(ks(1)) -abs(ks(1)) abs(ks(1))]), grid on;
xlabel('Kx');ylabel('Ky');zlabel('Kz');
set(gca,'Fontsize',12);

% Check to see which one of the shift of the k vector belongs to Kx, Ky,
% and Kz
ks(a);
ks(b);
ks(c);

% Create the Gaussian filter with width \tau = 1/2
g = exp(-((Kx-ks(b)).^2 + (Ky - ks(a)).^2 + (Kz-ks(c)).^2)/2);

% Apply the Gaussian filter to denoise the data
Path = [];
for j = 1:20
    Un(:,:,:) = reshape(Undata(j,:),n,n,n);
    Unt(:,:,:) = fftn(Un);
    Unts = fftshift(Unt);

    Unts_g = Unts.*g;
    Uns_g = ifftn(Unts_g);
    absUns_g = abs(Uns_g);

```

```
    cfreq = 0;
    for ii = 1:n
        for jj = 1:n
            for kk = 1:n
                if absUns_g(ii,jj,kk) > cfreq
                    cfreq = absUns_g(ii,jj,kk);
                    a = X(1,ii,1); b = Y(jj,1,1); c = Z(1,1,kk);
                end
            end
        end
    end
    % Store the indicies of all center frequencies which is the path of the
    % marble
    Path = [Path ; [b a c]];
end

% ------------------------------- Figure 3 ----------------------------------------
%
% Use plot3 to plot the path of the marble
figure(3)
plot3(Path(:,1),Path(:,2),Path(:,3),'-o','Color','k','LineWidth',2,'MarkerSize',13,'MarkerFaceColor','c');
axis([-20 20 -20 20 -20 20]), grid on;
xlabel('X');ylabel('Y');zlabel('Z');
set(gca,'FontSize',12);

% Print out the location of the marble at the 20th data measurement.
Path(20,:)

% ------------------------------- Figure 4 ----------------------------------------
%
% Plot the location of the marble at the 20th data measurement.
figure(4)
isosurface(X,Y,Z,absUns_g/cfreq,0.7);
axis([-20 20 -20 20 -20 20]), grid on;
xlabel('X');ylabel('Y');zlabel('Z');
set(gca,'FontSize',12);
```