AMATH 582 Homework 1

Reed Nomura

January 24, 2020

Abstract

This article explores an application of Fourier transforms to eliminate noise in data. In particular, this project begins with 20 data points collected from ultrasound of a dog's intestines. The purpose of this project is to eliminate the noise from the data points in order to locate a marble and compute its trajectory through the dogs intestines in order to target the marble with intense acoustic sound waves.

1 Introduction and Overview

The data provided is 20 different ultrasound measurements taken from a dog that swallowed a marble. Due to the dog's movement, the initial data is unintelligible. The goal of this article will be to make use of information in the frequency domain in order to eliminate the noise in the spatial domain and to track and locate the marble. The noisy data can be seen in Figure 1. Fast Fourier transforms will be applied to the data in the spatial domain in order to transform it into the frequency domain. In the frequency domain, the data can be averaged to reveal a centralized frequency (Figure 2) with which a Gaussian filter can be constructed. The filter will then be applied to the frequency data and inverse Fourier transformed back into the spatial domain. The data then will be free from noise and will show a clear trajectory of the marble through the dog's intestines (Figure 3) .

2 Theoretical Background

2.1 Fourier Series

Fourier introduced the concept of representing a given function $f(x)$ by a trigonometric series of sines and cosines: [1]

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

Through some basic mathematic manipulation, we are able to produce formulas for the Fourier coefficients.

$$a_n = \frac{1}{\pi} \int_{-\pi}^{\pi} f(x) \cos nx dx \tag{2}$$

$$b_n = \frac{1}{\pi} \int_{-\pi}^{\pi} f(x) \sin nx dx \tag{3}$$

The complex version of the expansion produces the Fourier series on the domain $x \in [-L, L]$ which is given by

$$f(x) = \sum_{-\infty}^{\infty} c_n e^{in\pi x/L} \quad x \in [-L, L]. \tag{4}$$

With the following Fourier coefficient.

$$c_n = \frac{1}{2L} \int_{-L}^{L} f(x) e^{in\pi x/L} dx \tag{5}$$

2.2 Fourier Transform

The Fourier Transform is an integral transform defined over the entire line $x \in [-\infty, \infty]$.[1] The Fourier transform is defined as

$$F(k) = \frac{1}{\sqrt{2\pi}} \int_{-\infty}^{\infty} e^{-ikx} f(x) dx \tag{6}$$

Thus the Fourier transform is essentially an eigenfunction expansion over all continuous wavenumbers k. And once we are on a finite domain $x \in [-L, L]$, the continuous eigenfunction expansion becomes a discrete sum of eigenfunctions and associated wavenumbers (eigenvalues)[1]. This concept has widespread applications. For the purposes of this paper, Fourier transforms will be used to transform data in the spatial domain into data in the frequency domain in order to average the signal and to create a filter for the data.

2.3 Averaging the Signal

White noise is normally distributed with a zero mean.Therefore, Over many signals, the white-noise should, on average, add up to zero.[1] The noise on the data in this paper is presumed to be white noise created from the movement of the dog during the time that the measurements were taken. Hence, when the data is in the Fourier domain, we will average the signal in order to potentially eliminate the white noise.

2.4 Filtering Data

Spectral filtering is a method which allows information to be extracted at specific frequencies.[1] In this paper, Gausian filtering will be used to focus our view in order to eliminate noise on the data. A Gausian filter

$$\mathcal{F}(k) = \exp -\tau(k - k_0)^2 \tag{7}$$

where τ measures the bandwidth of the filter, and k is the wavenumber. The generic filter function $\mathcal{F}(k)$ in this case acts as a low-pass filter since it eliminates high-frequency components in the system of interest. Note that these are high- frequencies in relation to the center-frequency ($k = k_0$) of the desired signal field[1]. Since our data is in three dimensional space, a filter will need to be applied to the central frequency on three axis.

2.5 Inverse Fourier Transform

Inverse Fourier transforms are used to undo a Fourier transform. The inverse Fourier Transform is defined as

$$f(x) = \frac{1}{\sqrt{2\pi}} \int_{-\infty}^{\infty} e^{-ikx} F(k) dk \tag{8}$$

For the purposes of this paper, inverse Fourier transforms will be used to move from the frequency domain back to the spatial domain after a filter has been applied.

3 Algorithm Implementation and Development

1. Load data
2. Perform FFT on data
3. Average data in frequency domain
4. Filter data
5. Perform IFFT
6. Record and Plot the data

Algorithm 1: Averaging the Data

Import data from `Testdata.mat`
for $j = 1 : 20$ **do**
  Extract measurement j from `Undata`
  Perform FFT on every dimension of each point j
end for
Average the transformed data to find the coordinate of the center frequency.

Algorithm 2: Filtering the Data

Begin with Fourier transformed data from Algorithm 1
Create a Gaussian filter centered around the center frequency found in Algorithm 1
for $j = 1 : 20$ **do**
  Extract measurement j from `Undata`
  Apply Gaussian filter and then perform IFFT each point j
end for

4 Computational Results

Initially the data was turned into a cube to be viewed. However, as seen in Figure 1 it was unintelligible.

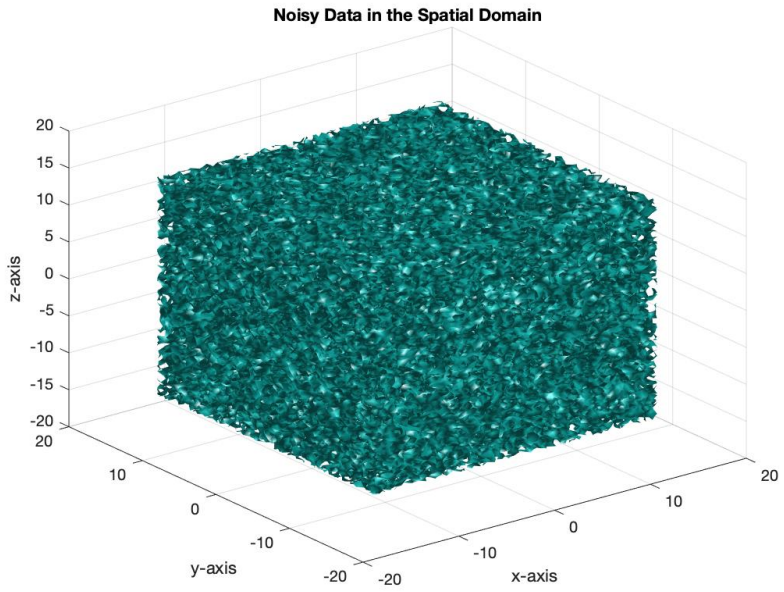

Figure 1: This is an image of the spatial data unfiltered

A Fourier transform was applied to the data. Since the noise on the data was suspected to be white noise, the data was averaged in order to eliminate it. Figure 2 shows this averaged signal in the Fourier domain.

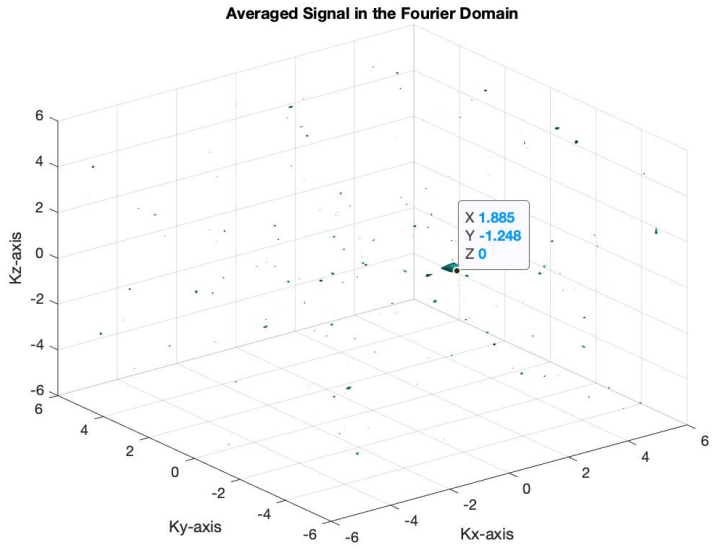

Figure 2: This is an image of the averaged data in the Fourier domain

There appeared to be a concentration of data shown in Figure 2. This central frequency on each axis was calculated to be:

Coordinate	Kx	Ky	Kz
	1.885	-1.248	0.00

A Gaussian filter was applied to the data in the Fourier domain around that central frequency. After the filter was applied, an inverse Fourier transform was applied to the data and it exhibited a clear trajectory of the marble through our coordinate system. Figure 3 displays the path of the marble from the first data point until the 20th.

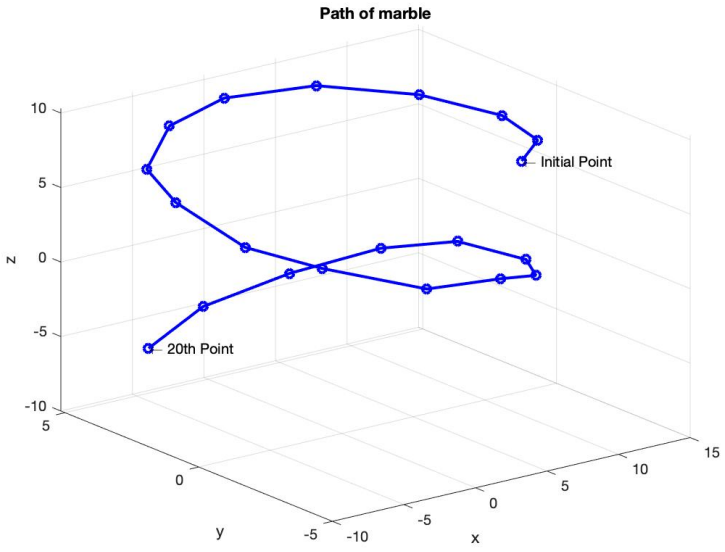

Figure 3: Here is an image of the path the marble took through the dog's intestines

The exact location of the marble in our coordinate system at each measurement can be seen here on Table 1.

Data Point	X	Y	Z
1	4.688	-4.219	10.312
2	8.438	-2.812	9.844
3	10.310	-0.469	9.375
4	8.906	1.875	9.375
5	6.094	4.219	8.906
6	1.406	5.156	8.438
7	-3.281	4.688	7.969
8	-7.500	3.281	7.031
9	-9.844	0.938	7.031
10	-9.375	-1.406	5.625
11	-7.500	-3.281	5.156
12	-2.812	-4.688	3.750
13	2.344	-4.688	3.281
14	6.562	-3.750	1.875
15	9.375	-1.875	0.938
16	9.844	0.938	0.000
17	7.969	2.812	-1.406
18	4.219	4.219	-3.281
19	-0.938	4.688	-4.688
20	-5.625	4.219	-6.094

Table 1: Location in space at each of the 20 measurements

As seen in Table 1, the exact location of the marble at the 20th data point is calculated to be at the following point in reference to our established coordinate system.

Data Point	X	Y	Z
20	-5.625	4.219	-6.094

5 Summary and Conclusions

The initial data was unintelligible due to a presence of white noise. By using a Fourier transform, the data was moved out of the spacial domain and into the frequency domain. By averaging the signals in the frequency domain, a clearer picture began to emerge. Averaging the signals in the frequency domain reduced the white noise and unveiled a frequency with which a Gaussian filter could be constructed. The filter was applied to the frequency data and inverse Fourier transformed back into the spatial domain. The data, now readable, showed a clear trajectory of the marble through the dog's intestines. Therefore, an intense acoustic wave should be aimed at the point x: -5.625, y: 4.219, z: -6.094 at the 20th data measurement in order to break up the marble and save the dog.

References

[1] Jose Nathan Kutz. *Data-driven modeling & scientific computation: methods for complex systems & big data*. Oxford University Press, 2013.

[2] *MathWorks Website*. URL: `https://www.mathworks.com/help/matlab/index.html`.

Appendix A MATLAB Functions

This code can be found at: https://github.com/ReedNomura/AMATH-582/blob/master/Homework1A.m

- `[X,Y,Z] = 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). [2]
- `B = reshape(A,sz1,...,szN)`: reshapes A into a sz1-by-...-by-szN array where sz1,...,szN indicates the size of each dimension. [2]
- `isosurface(X,Y,Z,V, isovalue)`: computes isosurface data from the volume data V at the isosurface value specified in isovalue. [2]
- `M = max(A)`: returns the maximum elements of an array. [2]
- `Y = fft(X)` computes the discrete Fourier transform (DFT) of X using a fast Fourier transform (FFT) algorithm. [2]
- `Y = fftn(X)`: returns the multidimensional Fourier transform of an N-D array using a fast Fourier transform algorithm. The N-D transform is equivalent to computing the 1-D transform along each dimension of X. The output Y is the same size as X. [2]
- Y = fftshift(X): rearranges a Fourier transform X by shifting the zero-frequency component to the center of the array. [2]
- `X = ifftn(Y)`: returns the multidimensional discrete inverse Fourier transform of an N-D array using a fast Fourier transform algorithm. The N-D inverse transform is equivalent to computing the 1-D inverse transform along each dimension of Y. The output X is the same size as Y. [2]
- `plot3(X,Y,Z)`: plots coordinates in 3-D space. [2]

Appendix B MATLAB Code

```
% AMATH 582 Homework 1
clear all; close all; clc;

% Load the data
load Testdata

%Setup
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); % Dimensions in Spatial Domain
[Kx,Ky,Kz]=meshgrid(ks,ks,ks); % Dimensions in Frequency Domain

% Visualize noisy data
for j=1:20
  Un(:,:,:)=reshape(Undata(j,:),n,n,n); % reshaping data into cube
end
figure(1)
  close all,
  isosurface(X,Y,Z,abs(Un),0.4)
  axis([-20 20 -20 20 -20 20]), grid on, drawnow
  title('Noisy Data in the Spatial Domain')
  xlabel('x-axis')
  ylabel('y-axis')
  zlabel('z-axis')
  pause(5)
% Figure out central frequency look in 1D

Unt_avg = 0;
    for j=1:20
   Un(:,:,:)=reshape(Undata(j,:),n,n,n); % Reshaping data into cube
   Unt = fftn(Un); % Transform data to Fourier domain
   Unt_avg = Unt_avg + Unt; % Add together each measurement in the Fourier Domain
    end
       % Get averaged signal in Fourier domain
   Unt_avg = Unt_avg./20;
   [M,linearInd] = max(abs(Unt_avg(:)));

   figure (2)
    close all;
    isosurface(Kx,Ky,Kz,fftshift(abs(Unt_avg)./max(abs(Unt_avg(:)))), 0.6)
    axis([-6 6 -6 6 -6 6]), grid on, drawnow
  title('Averaged Signal in the Fourier Domain')
  xlabel('Kx-axis')
  ylabel('Ky-axis')
  zlabel('Kz-axis')
  pause (5)
% Identify "peak" frequencies

%Components for the central frequency
[I,J,K] = ind2sub([n n n], linearInd);
Central_x = Kx(I,J,K);
Central_y = Ky(I,J,K);
Central_z = Kz(I,J,K);
Central_Frequency = sprintf('The central frequency in frequency domain is at %s %d ...
    %f.',Central_x,Central_y,Central_z);
disp(Central_Frequency)

% Create a filter and filter data
```

```
    % Create filter
    filter = exp(-1 * ((Kx - Central_x).^2 + (Ky - Central_y).^2 + (Kz - Central_z).^2));

    % Apply filter to data in Fourier domain

    position = zeros(3,20);
for j=1:20
    Un = reshape(Undata(j,:),n,n,n);
    Unft = fftn(Un) .* filter;
    Unf = ifftn(Unft);

    % Store coordinates of marble at each (time) step
    [~, ind] = max(abs(Unf(:)));
    [position_x, position_y, position_z] = ind2sub([n n n], ind);
    position(1,j) = X(position_x, position_y, position_z);
    position(2,j) = Y(position_x, position_y, position_z);
    position(3,j) = Z(position_x, position_y, position_z);
end

% Plot the marble's path
figure(3)
plot3(position(1,:), position(2,:), position(3,:),'b-o', 'linewidth', 2),
text(position(1,1), position(2,1),position(3,1),'\leftarrow Initial Point' );
text(position(1,20), position(2,20),position(3,20),'\leftarrow 20th Point' );
grid on;
title('Path of marble'), xlabel('x'), ylabel('y'), zlabel('z')

%% print the last position

Final_position = sprintf('The position of the marble at the 20th data point is x: %.3f, y: ...
    %.3f, z: %.3f',...
    position(1,20), position(2,20),position(3,20));

disp(Final_position)
```