Ultrasound analysis - Fourier transform

Hyesu Lee

January 24, 2020

Abstract

This report is to present the understanding of the Fourier transformation and its usage in frequency analysis. With given noised ultrasonic data, student is expected to practice data analysis by using Fourier transformation and Gaussian filter. As a result of data analysis, the student is able to report the exact location of the marble inside of a dog's intestines. This analysis is done by the MATLAB program.

1 Introduction and Overview

In modern days, data is used in various areas of study such as economy, physical science, technology, medical, education etc. Being able to analyze data and find meaningful information is evidently important skill. In this assignment, the student is asked to find the exact location of marble inside of a dog's intestine with 20 ultrasound data sets in different time frames. The purpose of the assignment is to filter raw data, frequency, by using Fourier transform and is to find the right trajectory of the marble based on the filtered data. By using Fourier transform, the process of finding the solution algorithm has low operation count and great accuracy.

2 Theoretical Background

2.1 Fourier series

Fourier introduced the idea that any given function and its derivation could be represented by a trigonometric series of sines and consines:

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

By multiplying *cosmx* on both sides and applying orthogonality property, Equation 2, Equation 3, Equation 4, the Fourier coefficients are derived. Fourier coefficients are Equation 5.

$$\int_{-\pi}^{\pi} \sin nx \cos mx \; dx = 0 \;\; \forall n, m \tag{2}$$

$$\int_{-\pi}^{\pi} \cos nx \cos mx \; dx = \begin{cases} 0 & \text{if } n \neq m. \\ \pi & \text{if } n = m. \end{cases} \tag{3}$$

$$\int_{-\pi}^{\pi} \sin nx \sin mx \; dx = \begin{cases} 0 & \text{if } n \neq m. \\ \pi & \text{if } n = m. \end{cases} \tag{4}$$

$$a_n = \frac{1}{L}\int_{-L}^{L} f(x)\cos\frac{n\pi x}{L}\; dx \;\; n \geq 0, \quad b_n = \frac{1}{L}\int_{-L}^{L} f(x)\sin\frac{n\pi x}{L}\; dx \;\; n > 0 \tag{5}$$

Since Equation 1 is composed of periodic functions cosine and sine, its expansion always lives in 2π period. Fourier series could be composed by complex numbers. Euler's identity, Equation 6, allows the series expansion to represent with complex number. Complex version of Fourier series is Equation 7 and its Fourier coefficient is Equation 8.

$$e^{\pm ix} = \cos x \pm i \sin x \tag{6}$$

$$f(x) = \sum_{-\infty}^{\infty} C_n e^{\frac{in\pi x}{L}} \quad x \in (-L, L] \tag{7}$$

$$C_n = \frac{1}{2L} \int_{-L}^{L} f(x) e^{\frac{-in\pi x}{L}} \quad dx \tag{8}$$

For Equation 7 and Equation 8, one could assume followings[**kutz˙2013**]:

- `$C_0$ is real and $C_{-n} = C_n$`
- `If $f(x)$ is even, all $C_n$ are real`
- `If $f(x)$ is odd, $C_0$ and all $C_n$ are purely imaginary`

Lastly, at a discontinuity of f(x), the Fourier series converges to the average of the left and right values of the discontinuity.

2.2 Fourier transform

Fourier transform is to represent a non-periodic function by a continuous superposition or integral of complex exponential. Equation 9 is Fourier transform. Equation 10 is inverse Fourier transform :

$$\mathcal{F}(k) = \frac{1}{\sqrt{2\pi}} \int_{-\infty}^{\infty} e^{-ikx} f(x) \quad dx \tag{9}$$

$$f(x) = \frac{1}{\sqrt{2\pi}} \int_{-\infty}^{\infty} e^{ikx} \mathcal{F}(k) \quad dk \tag{10}$$

This integral transform is defined over the entire line $x \in [-\infty, \infty]$
As shown in Equation 9, the formal definition covers real line $x \in [-\infty, \infty]$, but for the computational purpose, the computational domain is over finite domain $x \in [-L, L]$. Finite domain makes the Fourier transform to be a sum of eigenfunctions and its eigenvalues[**kutz˙2013**].
Fourier transform receives one function in specific domain such as time or spatial, then returns a function that lives in different domain, often frequency domain, without damaging data. Further, the simplicity of derivative relation of Fourier transform, Equation 11, makes itself useful.

$$\mathcal{F}(k) = \hat{f^n} = (ik)^n \hat{f} \tag{11}$$

2.3 Fast Fourier transform

Fast Fourier transform is a algorithm performs the forward and backward Fourier transforms in fast speed. Followings are the features of the Fast Fourier Transform(FFT)[**kutz˙2013**]:

- Low operation count $O(N \log N)$
- Finds transform on interval $x \in [-L, L]$ and due to integration kernel e^{ikx} is oscillatory, the solution on the finite interval have periodic boundary conditions
- The key to lowering the operation count is in discretizing the range $x \in [-L, L]$ into 2^n points
- The FFT has excellent accuracy properties, typically well beyond that of standard discretization schemes.

2.4 Filter

The first step for the data analyzation is to filter the data set such that only meaningful data remains. τ is bandwidth and k is wave number. Equation 12 is Gaussian filter in one dimensional space and Equation 13 is Gaussian filter for three dimensional space.

$$\mathcal{F}(k) = e^{-\tau(k-k_0)^2} \tag{12}$$

$$\mathcal{F}(k) = e^{-\tau((kx-kx_0)^2+(ky-ky_0)^2+(kz-kz_0)^2)} \tag{13}$$

3 Algorithm Implementation and Development

The algorithm is divided into three different parts: setting up the components, finding the filter, and filtering each data set for the marble location.

3.1 Setting up the spatial and frequency component

In order to use Fourier transformation, spatial and frequency component need to be defined. As given in the assignment prompt, spatial domain is (-15,15) and number of points for discretization is 64, 2^5, points. As explained in previous point, the number of points should be 2^n for FFT algorithm. Since the range is periodic bound, use linspace function with 65 and set up spatial axis with 64 values. The frequency component, k, needs to be multiplied by $\frac{2\pi}{2L}$, so that k lives in frequency range, 2π. Due to FFT algorithm implementation in MATLAB, variable ks, frequency component with correct order, needs to be defined with fftshift() function. Then, set up the 3-dimensional space with meshgrid function for both spatial and spectral space. See Algorithm 1 for MATLAB implementation.

Algorithm 1: Set up

```
    load Testdata
    Define the spatial range and number of points for discretization.
    Define the spatial space.
    Define x,y,z axis as vectors.
    Define frequency component as vector.
    Define another variable for the rightly ordered frequency component
    Define 3-D space with X, Y, Z for spatial component and Kx, Ky, Kz for the frequency component
```

3.2 Finding filter

It is known that white noise is modeled adding a normally distributed random variable with zero mean and unit variance to each Fourier component of the spectrum[**kutz˙2013**]. Having multiple data sets, it is possible to average the frequencies of 20 time frames and remove white noise. Since frequencies need to be added, the data set requires to be frequency domain. In order to compute complex number, absolute value of Fourier transform value is used for computation. The averaged value is the frequency of ultrasound which sent out to detect marble. This frequency is a filter to remove noise from each data set. The last information needed for the filter is its placement. The location of filter is found by finding the peak of averaged data set. Then, build the 3-D Gaussian filter with the filter location value. See Algorithm 2 for further explanation.

Algorithm 2: Finding filter

```
load Testdata
for $j = 1:20$ do
    Extract measurement $j$ from Undata
    Fourier transform the data
    Cumulatively add the data into variable, total.
end for
Find the average of the Fourier transformed frequencies
Find the highest point of the average dataset and save its axis location as variables
Convert the three axis location from spatial domain to frequency domain.
Build Gaussian filter with the location found
```

3.3 Applying filter

With the filter, the data set shows more clear visualization of the situation. To apply filter, filter needs to be multiplied to each data set. Since the filtering is done in frequency domain, each data set needs to be Fourier transform. Whenever data set Fourier transform using fft() command, the order of data set get suffled. Therefore, use MATLAB function fftshift() and ifftshift() accordingly. Then, use same algorithm used for finding filter location to discover location of marble. See Algorithm 3 for further implementation detail.

Algorithm 3: Applying filter

```
for $j = 1:20$ do
    Extract measurement $j$ from Undata
    Fourier transform the data and use fftshift() if necessary
    Multiply the data set with filter
    Inverse Fourier transform the data, use ifftshift() if necessary
    Find the index of the highest value in the data set
    Save the location values as matrix for plotting
end for
Plot the trajectory of marble using the matrix found.
```

4 Computational Results

Add your computational results here. See Table 1 for each time frame x, y, z axis location. See Figure 1 and Figure 2 for the trajectory of the marble. In the 20th time frame, marble is placed in (-5.625, 4.218, -6.093).

5 Summary and Conclusions

By using Fourier transformation, the 20 ultrasound data set could be filtered and present the location of marble inside of the dog, fluffy. In the 20th time frame, marble is placed in (-5.625, 4.218, -6.093).

	x-axis	y-axis	z-axis
1	4.687	-4.218	10.312
2	8.437	-2.812	9.843
3	9.843	-0.468	9.843
4	8.906	2.343	9.375
5	6.093	4.218	8.906
6	1.406	5.156	8.437
7	-3.281	4.687	8.437
8	-7.500	3.281	7.500
9	-9.843	0.937	7.031
10	-9.843	-1.406	5.625
11	-7.031	-3.281	5.156
12	-2.812	-4.687	4.218
13	2.343	-4.687	3.281
14	6.562	-3.750	2.343
15	9.375	-1.875	1.406
16	9.843	0.468	-0.468
17	7.968	2.812	-1.406
18	4.218	4.218	-2.812
19	-0.937	4.687	-4.218
20	-5.625	4.218	-6.093

Table 1: Location of marble in each time frame

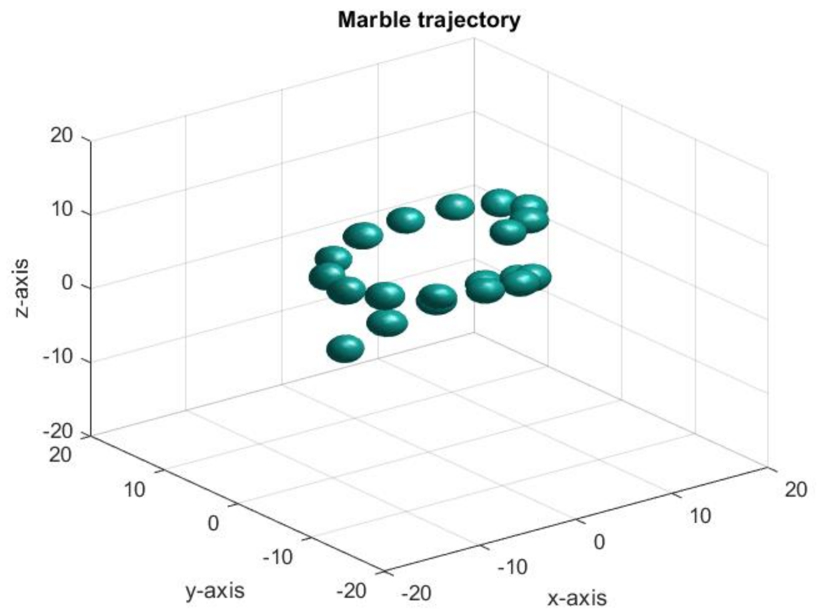

Figure 1: Trajectory of marble plotted by isosurface function

Appendix A MATLAB Functions

changing x and y using ind2sub Followings are MATLAB Functions used for the assignment:

- `y = linspace(x1,x2,n)` returns a row vector of `n` evenly spaced points between `x1` and `x2`.
- `y = fftshift(x)` return the rearranged Fourier transform x which contains zero-frequency compo-

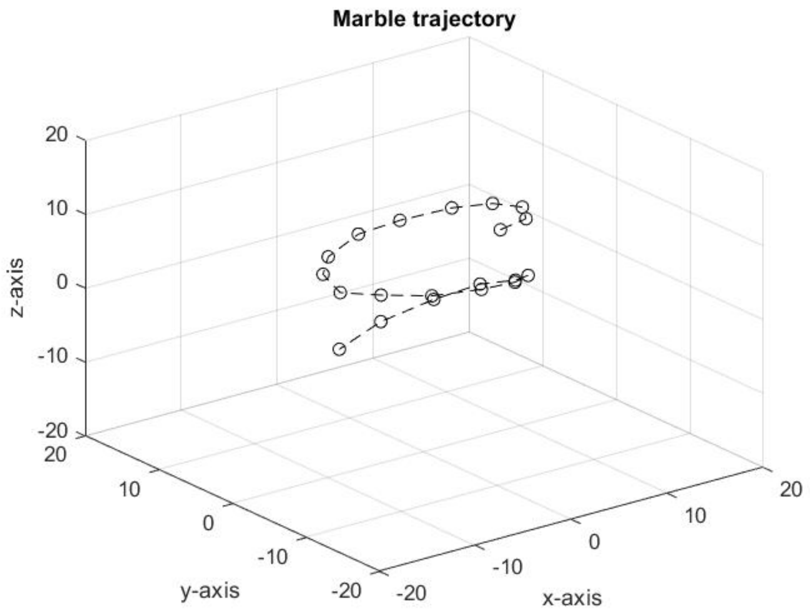

Figure 2: Trajectory of marble plotted by Plot3 function

nent at the center of the array.

- `y = fftn(x)` returns multidimensional Fourier transform of nth dimensional array using a fast Fourier transform algorithm. It needs fftshift() for rearrangement.
- `B = reshape(A, sz)` return reshaped matrix A, using the size vector sz for size of matrix B.
- `[y, x, z] = ind2sub(sz, ind)` returns three arrays containing equivalent multidimensional subscripts corresponding to the linear indices ind for a multidimensional array of size sz. Since the return value format is [row,col], assignment for x-axis and y-axis should be ordered as [y-axis, x-axis].
- `x = ifftshift(y)` return rearranged a zero-frequency-shifted Fourier transform y back to the original transform output.
- `x = ifftn(y)` returns the multidimensional inverse Fourier transform of an nth dimensional array using a fast Fourier transform algorithm.
- `isosurface(x, y, z, value, isovalue)`creates a graph in the given axes with the computed faces and vertices. In implementation, value is norm of the data for plot.
- `plot3(x, y, z, LineSpec)` plots coordinates in 3-D space with specific line style.
- `[X,Y] = meshgrid(x,y)` returns 2-D grid coordinates based on the coordinates contained in the vectors `x` and `y`. X is a matrix where each row is a copy of `x`, and `Y` is a matrix where each column is a copy of `y`. The grid represented by the coordinates `X` and `Y` has `length(y)` rows and `length(x)` columns.

Appendix B MATLAB Code

```
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]; % frequency components of FFT
ks=fftshift(k);% unshifted counter parts

[X,Y,Z]=meshgrid(x,y,z);
[Kx,Ky,Kz]=meshgrid(ks,ks,ks);
total = zeros(n,n,n);

% finding filter by taking average
for j=1:20
    Un(:,:,:)=reshape(Undata(j,:),n,n,n);
    total = total+fftn(Un);
end
signal = abs(fftshift(total))/20;

[val,idx] = max(abs(signal(:)));
[fx,fy,fz] = ind2sub([64,64,64], idx);
kx = ks(fx); % fourier domain filter frequency placement
ky = ks(fy); % fourier domain filter frequency placement
kz = ks(fz); % fourier domain filter frequency placement
tau = 2; % bandwidth
filter = exp(-tau*((Kx - ky).^2+(Ky - kx).^2+(Kz - kz).^2));
layer = [0,0,0];

for j = 1:20
    % filtering data in frequency domain.
    Un(:,:,:)=reshape(Undata(j,:),n,n,n);
    Unt = fftshift(fftn(Un));
    Untf = filter .* Unt;
    Untfp = ifftn(ifftshift(Untf));
    figure(1);
    isosurface(X,Y,Z,abs(Untfp)./max(abs(Untfp(:))),0.8)
    title('Marble trajectory');
    xlabel('x-axis'), ylabel('y-axis'), zlabel('z-axis');
    axis([-20 20 -20 20 -20 20]), grid on, drawnow
    hold on
    [val,indx] = max(abs(Untfp(:)));
    [yidx, xidx, zidx] = ind2sub([n,n,n], indx);
    x = Y(xidx,yidx, zidx);
    y = X(xidx,yidx, zidx);
    z = Z(xidx,yidx, zidx);
    layer = [layer; [x y z]];
end

figure(2);
layer = layer(2:end, :);
plot3(layer(:,1),layer(:,2),layer(:,3),'--ko');
axis([-20 20 -20 20 -20 20]), grid on
title('Marble trajectory');
xlabel('x-axis'), ylabel('y-axis'), zlabel('z-axis');
```

Appendix C Github page

https://github.com/HyesLee99/Data-analysis.git