Ruimin Zhang
Jan 24th, 2020

Ultrasound Diagnostic Using Fourier Transformation

Abstract:

This is the homework 1 of AMATH 482 in the University of Washington. We use FFT, averaging, and filtering to find our wanted signal graph over time.

Sec. I. Introduction and Overview

The problem is to save a dog who swallowed a marble. The vet used ultrasound and obtained data in a small area of the intestines. However, the data is highly noised. My job is to denoise the data and help the doctor locate where the marble is.

Sec. II. Theoretical Background

2.1 The Fourier Transform

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

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

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

Using the knowledge of Fourier transformation, we are able to approximate a continuous, periodic function on a finite domain x ∈ [-L, L] by a discrete sum of eigenfunctions and associated eigenvalues.

2.2 Fast Fourier transform

The Fast Fourier transform (i.e. FFT), even though using the same strategy, is faster than performing Fourier transform. The operation count drops to O(N log N) once we apply the FFT. The basic algorithm behind FFT is that we discretize the range N into 2^n points. Each of the 2^n point is called a Fourier mode.

We first consider the time domain that we want to perform FFT over. Besides transform original function, the FFT comes with three other properties:

1. It shifts data such that x ∈ [0, L] → [−L, 0] and x ∈ [−L, 0] → [0, L], and
2. It multiplies every other mode by −1, and
3. It assumes you are working on a 2π periodic domain so that scale our frequency by $\frac{2\pi}{L}$

The first two processes are called aliasing, and the last one is rescaling. Because of the aliasing, we need to shift our resulting frequency graph after applying FFT so that we can transform the function back to the mathematically correct positions. Once we use shifting on FFT, we can use inverse shifting to go back to our unshifted data. Similarly, we can always use inverse FFT to find the untransformed data. Shifting FFT is very important in graphing.

2.3 Averaging and Filtering

Since the ultrasound data we received is highly noised, we need to denoise and filter the data so that we can locate the position of the marble. Averaging is used to find central frequency when we have too much noise in our data. Filtering is used when given a central frequency k_0, we can enhance the region around k_0 and undermine the rest region. In this example, the noise is Gaussian distributed, meaning the mean of noises is 0, so we use averaging to denoise. Additionally, we choose to apply the Gaussian filter in our case. The filter we use is

$$g = e^{-a(k-k_0)^2}$$

where a is the width of the filter and k is a vector.

Sec. III. Algorithm Implementation and Development

3.1 Basic algorithm

In order to solve this problem, we need to analyze our data and find the location of the marble. To do this, we proceed five through steps:

(i) Sort the data
(ii) Denoise our data to find central frequency
(iii) Filter data around central frequency
(iv) Find the path of the marble with more accuracy
(v) The end of the path is where the marble is

For (i), we first load our data, create the spatial domain and Fourier modes, and reshape our data into 20 different 64×64×64 matrices, each is called a Un matrix with size 64×64×64. We observe that 64 = 2^6, so we can create 2^6 Fourier nodes on each of the x, y, z axis.

3.2 Denoise

The main idea is we take the sum of transformed matrices and divide it by the number of matrices. The specific steps are stated below:

(i) Sum up the FFT values of the 20 Un matrices
(ii) Take the absolute value. Because FFT gives us both real and imaginary values, we take the absolute value of the FFT sum in (i) to observe the real values.
(iii) Eliminate noise. We take the value from (ii) and divide the number of matrices to find the average.
(iv) Scale down. Divide the average value by its maximum value such that we can have all the values within 1.
(v) Apply FFT shift before drawing since we want to find the central frequency through a graph.

(vi) Finally, we ask MATLAB to draw a graph. After scaling our axis, we are able to locate the central frequency k_0, which is the marble frequency.

3.3 Filter

Now that we have the central frequency k_0, the next step is to apply the filter function.

(i) Define parameters k_0, a, and filter function $g = e^{-a(k-k_0)^2}$

(ii) Take each of the 20 transformed Un matrices and multiply with g. Now we have the transformed filtered function, call it Sg (Sg is in frequency domain).

(iii) Apply the inverse FFT, take the absolute value of it, and divide by its maximum. The resulting function, call it Sgt, is in the time domain. Sgt represents the signal of the marble during a specific time period.

(iv) Graph each of the 20 Sgt over time domain, and we can see the motion of the marble.

As mentioned before, the parameter a determines the width of the function. If we increase the value of a, the size of the ellipsoid (i.e. enhanced region around our signal) is increasing. This means the width increased as a increases; and vice versa.
We observed that the position is shown as ellipsoids which is not very accurate. So, we can perform one more step to add precision:

(v) Find the maximum value within each ellipsoid, which is a more precise position of the marble. Then we draw a graph to show the locations of the corresponding maximum signals. This graph shows the path of the marble.

3.4 Locate the final position of the marble

Then our final step is to tell the doctor ASAP that the location of the marble is at the end of the path.

Sec. IV. Computational Results

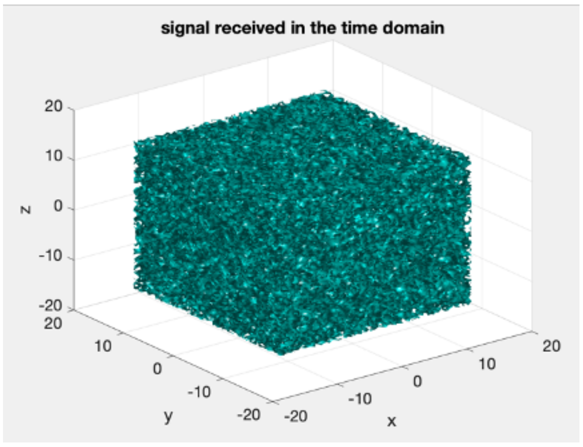

Figure 1. Data collected using ultrasound. We notice the graph is highly-noised.

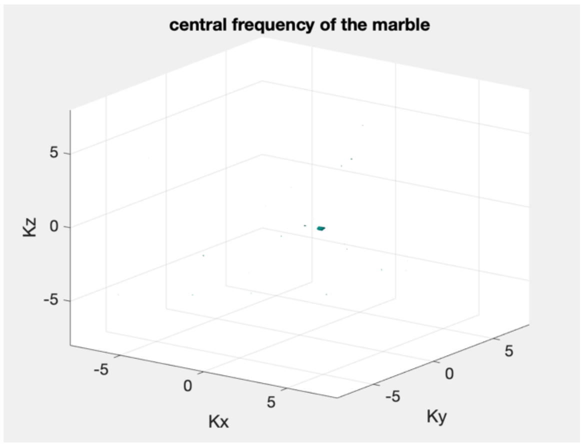

Figure 2. Central frequency after denoising. Once we average the FFT values, an obvious spot appears which is the central frequency we are looking for. By clicking it, we can get the position of this frequency, which is around (2.094, -1.047, 0).

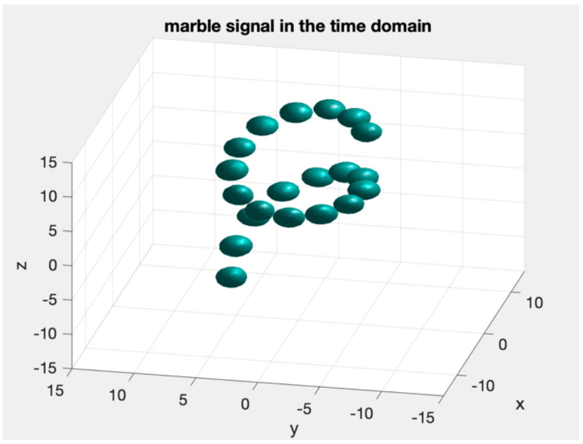

Figure 3. Untransformed filtered data. This graph is obtained by processing the first 4 steps of #3.3. By the filter function, we are able to focus on specific region around the central frequency. This is also the region where the marble lies within. This graph roughly shows us the motion of the marble.

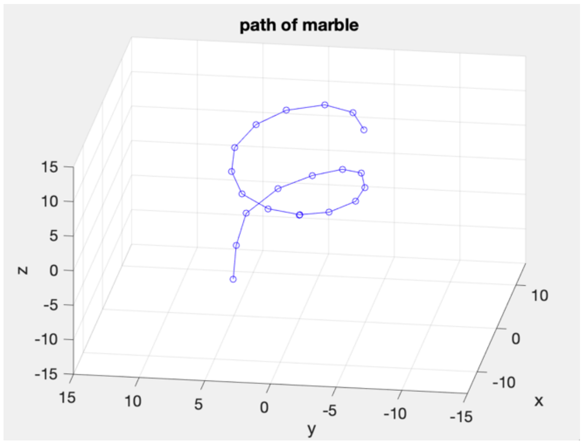

Figure 4. Path of the marble. This graph is obtained by processing step (i) through (iii) and (v) in #3.3. We locate the position of the marble with higher precision. The end of the path (also the lowest point) is the final location of the marble.

Sec. V. Summary and Conclusions

To sum up, the most important technique used in this problem is the fast fourier transformation to transform data in the time domain to the frequency domain. We locate the central frequency by denoise the data (through averaging). Then we use the central frequency to create a filter function. Combining the filter function to our data, we are able to focus on the data near the central frequency and ignore the unwanted data. Finally, we locate the precise position of the marble.

Appendix A. MATLAB functions used and brief implementation explanation

1. B = reshape(A,sz1,...,szN) reshapes A into a sz1-by-...-by-szN array where sz1,...,szN indicates the size of each dimension. You can specify a single dimension size of [] to have the dimension size automatically calculated, such that the number of elements in B matches the number of elements in A. For example, if A is a 10-by-10 matrix, then reshape(A,2,2,[]) reshapes the 100 elements of A into a 2-by-2-by-25 array.

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.

3. Y = fftshift(X) rearranges a Fourier transform X by shifting the zero-frequency component to the center of the array.

4. M = max(A) returns the maximum elements of an array.

5. [M,I] = max(___) also returns the index into the operating dimension that corresponds to the maximum value of A for any of the previous syntaxes.

6. fv = isosurface(X,Y,Z,V,isovalue) computes isosurface data from the volume data V at the isosurface value specified in isovalue. That is, the isosurface connects points that have the specified value much the way contour lines connect points of equal elevation.

7. 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.

8. X = zeros(n) returns an n-by-n matrix of zeros.

9. plot3(X,Y,Z) plots coordinates in 3-D space.

Appendix B.MATLAB codes

```
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); % Kx, Ky, Kz are all 64*64*64 size

ave = 0;
for j=1:20
    Un(:,:,:)=reshape(Undata(j,:),n,n,n);
    ave = ave + fftn(Un);
end
ave = abs(fftshift(ave)/20);

isosurface(Kx,Ky,Kz,ave/max(max(max(ave))),0.7);
hold on
axis([-8 8 -8 8 -8 8]), grid on, drawnow
xlabel('Kx')
ylabel('Ky')
zlabel('Kz')
title('central frequency of the marble')
set(gca,'Fontsize',15)

%%
% we observed that the central frequency is (2.094,-1.047,0)
% now we consider applying the filter function around the central frequency
close all;
tau = [2.094 -1.047 0]; % central frequency
a = 1; % width of the filter

% calculation of frequency vector in 3D
g = exp(-a*((Kx-tau(1)).^2+ (Ky-tau(2)).^2+ (Kz-tau(3)).^2));

for j=1:20
    Un(:,:,:)=reshape(Undata(j,:),n,n,n);
    Sg = fftshift(g).*fftn(Un);
    Sgt = ifftn(Sg);
    Sgt = abs(Sgt)./max(max(max(abs(Sgt))));
    isosurface(X,Y,Z,Sgt,0.8),hold on
    axis([-15 15 -15 15 -15 15]), grid on, drawnow
end
xlabel('x')
ylabel('y')
zlabel('z')
title('marble signal in the time domain')
set(gca,'Fontsize',15)

%%

close all;
```

```
position = 0;
x1 = zeros(20);
y1=zeros(20);
z1=zeros(20);
for j=1:20
    Un(:,:,:)=reshape(Undata(j,:),n,n,n);
    Sg = fftshift(g).*fftn(Un);
    Sgt = ifftn(Sg);
    Sgt = abs(Sgt)/max(max(max(abs(Sgt))));
    [M, I] = max(Sgt(:));
    x1(j) = X(I);
    y1(j) = Y(I);
    z1(j) = Z(I);
end
plot3(x1,y1,z1,'bo-'), hold on
axis([-15 15 -15 15 -15 15]), grid on, drawnow
xlabel('x')
ylabel('y')
zlabel('z')
title('path of marble')
set(gca,'Fontsize',15)
```