# Homework1: An ultrasound problem

Hengji Wang

January 16th, 2019

## Abstract

This report represents a method of denoising an ultrasound dataset using fast Fourier transform (FFT) and a Gaussian filter. The noisy data which consist twenty different measurement is collected from a small area of the intestines containing a marble. The frequency signature of the marble is located in frequency domain by averaging the spectrum. A Gaussian filter is applied to denoise the spectrum data. Positions of the marble in different measurements are obtained by inverse transform the denoised spectrum to the coordinate space. Trajectory is plotted by sequencing the twenty positions and the position of the marble at the last measurement is identified for treatment with an intense acoustic wave.

## 1 Induction and Overview

Fast Fourier transform (FFT) is a widely used method in signal processing and data analysis. Dataset which is noisy in time domain may have a relative distinguishable frequency signature in frequency domain, which allows us to filter the interested signal out with some denoising methods. In this problem, we have a set of twenty ultrasound measurements from the intestine of a dog, which contains an swallowed marble. The data are noisy because of the movement of the dog and the internal fluid in it. To save that dog, we have to find a way to locate and compute the trajectory of the marble, and then report the position of it at the last measurement so that we can apply an intense acoustic wave to breakup the marble.

Next sections will present a method that includes FFT algorithm, spectrum averaging and Gaussian filtering. Theoretical concepts and algorithms will be explained in the next section. Then computational results will be reported and plotted. MATLAB functions and codes will be appended at the end.

# 2 Theoretical Background

## 2.1 Fourier Tranform

Fourier transform is a tool that decomposes a function into an alternate representation. For $x \in (-\infty, \infty)$, the Fourier transform of a 1-D function $f(x)$ is

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

For a multidimensional function $f(x_1, x_2, \cdots, x_m)$, the Fourier transform is

$$\hat{f}(k_1, k_2, \cdots, k_m) = (\frac{1}{\sqrt{2\pi}})^m \int_{-\infty}^{\infty} f(x_1, x_2, \cdots, x_m) e^{-ik_1x_1} e^{-ik_2x_2} \cdots e^{-ik_nx_n} dx_1 dx_2 \cdots dx_m \tag{2}$$

The inverse transform is

$$f(x_1, x_2, \cdots, x_m) = (\frac{1}{\sqrt{2\pi}})^m \int_{-\infty}^{\infty} \hat{f}(k_1, k_2, \cdots, k_m) e^{ik_1x_1} e^{ik_2x_2} \cdots e^{ik_nx_n} dk_1 dk_2 \cdots dk_m \tag{3}$$

For this problem $x, y, z \in [-L, L]$ are divided into $n$ modes, which means the transformation should be a discrete sum.

## 2.2 FFT

FFT is an algorithm that computes the Fourier transform in a time complexity of $O(n \log n)$, which is realized by recursively dividing the Fourier function into two smaller ones, and thus it has a restriction that requires the domain to be dicretized into $2^n$ points. Corresponding functions have been constructed in MATLAB such as *fft, fftn, ifft, ifftn* etc., which will be introduced in next sections.

## 2.3 Spectrum Averaging

Averaging is a powerful method for denoising a set of noisy signals from different measurements. Since the sum of the white-noise should be zero, by averaging the signals the real interested signal will become more distinguishable, and thus the central frequency of the spectrum can be located.

## 2.4 Gaussian Filtering

A spectrum filter around the central frequency can be used to suppress the unintrested areas and thus filter out the remain components in which we are interested. Gaussian filter is the most common one, which is easy to apply by multiplying the spectrum data with a 1-D or multidimensional Gaussian function which is centered at the central frequency of data.

# 3 Algorithm Implementation and Development

- Load data from **Testdata.mat**
- Define axes of spatial domain and frequency domain
  - Divide $x, y, z \in [-L, L]$ into $n$ discrete modes as the axes in spatial domain
  - Obtain the axes of frequency domain by multiplying the modes in $[-\frac{n}{2}, \frac{n}{2})$ with $\frac{2\pi}{L}$ because FFT functions in MATLAB assume signals are $2\pi$ periodic. Also the axes should be shifted in an order of $[0, k_{\max}] \to [k_{\min}, 0]$ because the FFT functions in MATLAB will shift the data in that way.
- Construct coordinate systems using *meshgrid*
- Reshape the data into 3D using *reshape*
- Do FFT using *fftn* on the 20 measurements and take average by add them together into **Untave**
- Compute matrix **specUntave** where each element is the absolute value of the corresponding element in **Untave**, normalize it by dividing it with its largest element.
  - Taking the absolute value is because **Untave** is a complex matrix and the absolute values represent the intensity of signal
- Locate the central frequency using *find*
- Construct a 3D Gaussian filter around the central frequency
- Multiply each spectrum data **Unt** with the Gaussian filter and obtain **unft** at each measurement
- Inverse transform each **unft** into spatial domain as **unf** using *ifftn*
- Locate the element of each **unf** with the largest absolute value; store its position
- Plot the positions using *plot3*
- Report the position of the marble at the last step

# 4 Computational Results

The isosurface visualization (with isovalue 0.8) of the data at the first measurement is plotted in Figure 1, which is very noisy to distinguish the signal of marble.

The isosurface visualizations (with isovalue 0.6) of the data in the frequency domain before and after averaging are plotted in Figure 2. The central frequency was computed as $[k_x, k_y, k_z] = [1.885, -1.047, 0]$

The isosurface visualizations (with isovalue 0.8) of all positions after denoising and filtering and the corresponding trajectory plotted by *plot3* are plotted in Figure 3.

The final position of the marble is computed as $x = -5.625, y = 4.219, z = -6.094$.

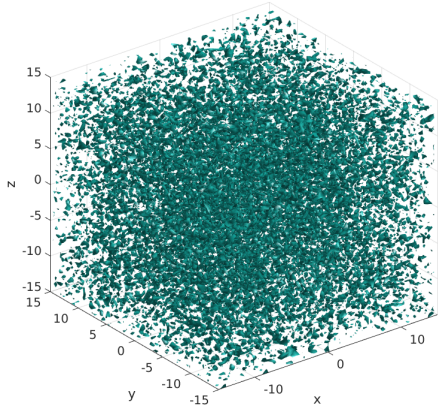


Figure 1: Isosurface visualization for the 1st measurement in spatial domain

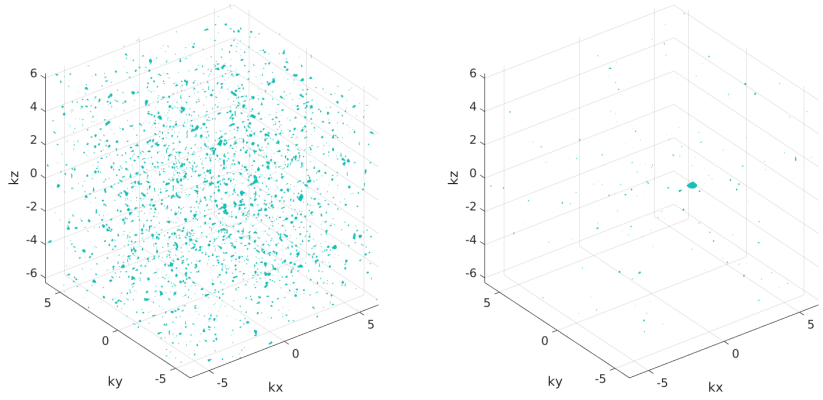


Figure 2: Isosurface visualizations in the frequency domain for the 1st measurement (left) and the averaged spectrum (right)

# 5 Summary and Conclusions

Spectrum averaging successfully helped to locate the central frequency. The method combining spectrum averaging and Gaussian filtering successfully helped us denoise the data and locate the marble in every measurement. The final position of the marble is at $(-5.625, 4.219, -6.094)$, where an intense acoustic wave should be focused.

# Appendix A: MATLAB functions used

- *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.

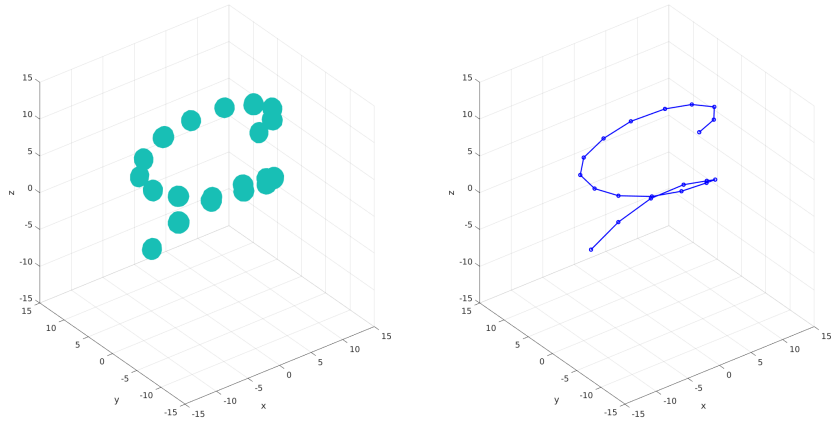

Figure 3: Isosurface visualizations in the spatial domain for the positions after denoising (left) and the trajectory (right)

- *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,...,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.
- *[row,col] = find(X)*: Returns the row and column subscripts of each nonzero element in array X.
- *isempty(A)*: Returns logical 1 (true) if A is an empty array and logical 0 (false) otherwise.
- *zeros(sz1,...,szN)*: Returns an sz1-by-...-by-szN array of zeros where sz1,...,szN indicate the size of each dimension.
- *ifftn(Y)*: Returns the multidimensional discrete inverse Fourier transform of an N-D array using a fast Fourier transform algorithm.
- *plot3(X1,Y1,Z1,...)*: 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

```
%% load data
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]; ks=fftshift(k);

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

% plot the noisy data of measurement 1
figure(1)
Un1(:,:,:)=reshape(Undata(1,:),n,n,n);
isosurface(X,Y,Z,abs(Un1),0.8)
axis([-L L -L L -L L]), grid on, drawnow
xlabel('x'); ylabel('y');zlabel('z');
pause(1)

%% do fourier transformation on data and average the spectrum
Untave = 0;

for j=1:20
    Un(:,:,:)=reshape(Undata(j,:),n,n,n);
    Unt = fftn(Un);
    Untave = Untave + Unt;
end

%% locate the central frequency

% normalize spectrum 1 (for comparison)
specUnt1 = abs(fftn(reshape(Undata(4,:),n,n,n)));
spec1d1 = reshape(specUnt1, n^3, 1);
specmax1 = max(spec1d1);
specUnt1 = specUnt1/specmax1;

% normalize averaged spectrum
specUntave = abs(Untave);
spec1d = reshape(specUntave, n^3, 1);
specmax = max(spec1d);
specUntave = specUntave/specmax;

% plot the isosurface in frequency domain
figure(2)

subplot(1,2,1)
isosurface(Kx,Ky,Kz,fftshift(specUnt1),0.6)
axis([-2*pi 2*pi -2*pi 2*pi -2*pi 2*pi]), grid on, drawnow
xlabel('kx'); ylabel('ky');zlabel('kz');
pause(1)

subplot(1,2,2)
```

```
isosurface(Kx,Ky,Kz,fftshift(specUntave),0.6)
axis([-2*pi 2*pi -2*pi 2*pi -2*pi 2*pi]), grid on, drawnow
xlabel('kx'); ylabel('ky');zlabel('kz');
pause(1)

% search the central frequency (with the biggest signal intensity)
for m = 1:n
    [i,j] = find(specUntave(:,:,m) == 1);
    if isempty(i)≠1 && isempty(j)≠1
        cent_freq = [i, j, m];
        break
    end
end

%% make a filter
filter = exp(-1.0*(fftshift(Kx)-k(cent_freq(2))).^2) .* ...
    exp(-1.0*(fftshift(Ky)-k(cent_freq(1))).^2) .* ...
    exp(-1.0*(fftshift(Kz)-k(cent_freq(3))).^2);

%% compute and plot the trajectory

position = zeros(20, 3);

figure(3) % for plotting the isosurface in spatial domain

subplot(1,2,1)
for j = 1:20
    % apply filter on each spectrum and ifft back
    Un(:,:,:)=reshape(Undata(j,:),n,n,n);
    Unt = fftn(Un);
    unft = filter.* Unt;
    unf = ifftn(unft);

    % get the value of strongest signal
    locunf = abs(unf);
    loc1d = reshape(locunf, n^3, 1);
    locmax = max(loc1d);

    % locate the strongest signal
    for locz = 1:n
        [locy,locx] = find(locunf(:,:,locz) == locmax);
        if isempty(locx)≠1 && isempty(locy)≠1
            position(j,:) = [x(locx), y(locy), z(locz)];
            break
        end
    end

    isosurface(X,Y,Z,locunf/locmax,0.8)
    axis([-L L -L L -L L]), grid on, drawnow
    xlabel('x'); ylabel('y');zlabel('z');
    pause(1)

end
```

```
% plot the trajectory
subplot(1,2,2)
plot3(position(:,1), position(:,2), position(:,3), 'bo-', 'Linewidth', 2)
axis([-L L -L L -L L]), grid on, drawnow
xlabel('x'); ylabel('y');zlabel('z');

%% print the last position

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

disp(result)
```