Application of Fast Fourier Transform in Ultrasound Data

Htet Aung Myin

January 24, 2020

Abstract

A dog has swallowed a marble. Extremely noisy data caused by internal and external factors are obtained through ultrasound. Through applications of signal averaging, the Fast Fourier Transform and the Gaussian filter, the signal will be filtered and de-noised. The trajectory is plotted and the final location of the marble is to be marked to be destroyed.

1 Introduction and Overview

The problem given is regarding a dog, Fluffy, who swallowed a marble. As the marble travel through the dog's intestines, data is obtained using ultrasound. As the dog is moving along with internal fluid movement, extremely noisy data is generated. The goal is to locate and compute the trajectory of the marble in order to use acoustic waves to breakup the marble.

2 Theoretical Background

2.1 Fast Fourier Transfrom

A Fast Fourier Transform(FFT) is an algorithm that computes the Discrete Fourier Transform (DFT) of a sequence. The Fourier Transforms converts a signal from the spatial domain into a frequency domain, or vice versa. The DFT is defined by the formula:

$$X_k = \sum_{n=0}^{N-1} x_n e^{-2\pi ikn/N} \quad k = 0, ..., N-1$$

while the inverse Fourier Transform (iFFT) defined by:

$$X_n = \frac{1}{N} \sum_{k=0}^{N-1} x_k e^{2\pi ikn/N} \quad k = 0, ..., N-1$$

with $N = 2^n$ In comparison, the FFT runs at $O(N \log N$ while the DFT runs at $O(N^2)$

2.2 Averaging

Averaging is a technique applied to data to increase the strength of a signal relative to the noise in the frequency domain. This process only works with white noise. As the time period increases, the noise averages out into obscurity while the signal comes into prominence.

2.3 3D Gaussian Filter

A 3D Gaussian Filter is defined as

$$F(k) = e^{-\tau(k_x - k_x 1)^2 + (k_y - k_y 1)^2 + (k_y - k_y 1)^2}$$

Applying a Gaussian Filter results in increasing the significance of the center or desired frequency while decreasing the rest. This decreases the noise and amplifies the signal desired.

3 Algorithm Implementation and Development

The data, containing 262144 values, with file name Testdata.mat is first loaded. Since the Fast Fourier Transform assumes 2π periodic signals, the frequencies are re-scaled by $2\pi/(2L)$ where L is the limits of the domain as described in line 7 of Appendix B. (Note: Any line references will be referring to Appendix B). Frequency values are stored in array k while the shifted frequency, for more convenient plots are stored in array ks. As the data is in 3D space, a meshgrid of X,Y, and Z coordinates are created for the spatial domain and Kx, Kz, Kz are created for the frequency domain.

The n-dimensional Fast Fourier Transform is applied to the data to move the data from the spatial domain to the frequency domain. Once the data is moved into the frequency domain, an averaging procedure over the twenty time period is applied to reduce noise and identify the marble. This is applied in lines 29-33. As noise averages out to insignificant values over the time period while the marble remains significant, coordinates of the marble can be found where the maximum signal is detected as described in lines 37-38, 54-55.

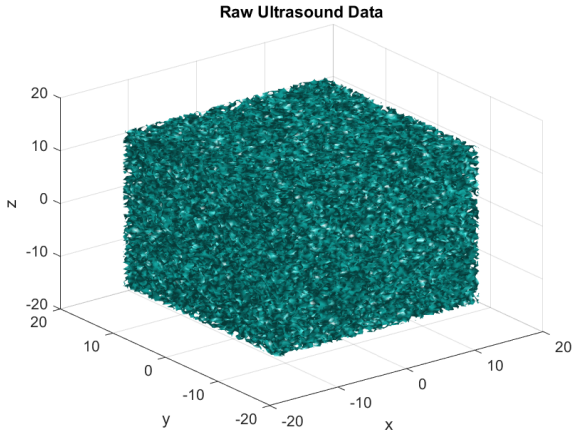
After the center frequency signal of the marble is detected, a 3D Gaussian filter is applied to denoise further. A meshgrid of unshifted frequency domains are declared in order to calculate the filter function. The filter function is defined on line 66. The filter function is then applied over to the Fast Fourier Transform of the raw data, declared as utn in line 70, over a course of the 20 time period, as shown in line 71. The filtered data is then subjected to the n-dimensional inverse Fast Fourier Transform in line 72 and the maximum signal of each time, corresponding to the marble's center in data is then converted into coordinates of the marble at its given time and stored into an array in line 74-75.

The final location of the marble is then determined by taking the final value of the array.

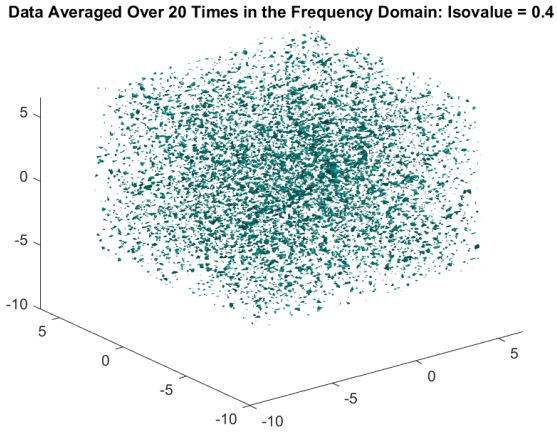
Plots were generated in lines 15-21(Figure 1), 39-48(Figure 2,3), 79-83(Figure 4)

4 Computational Results

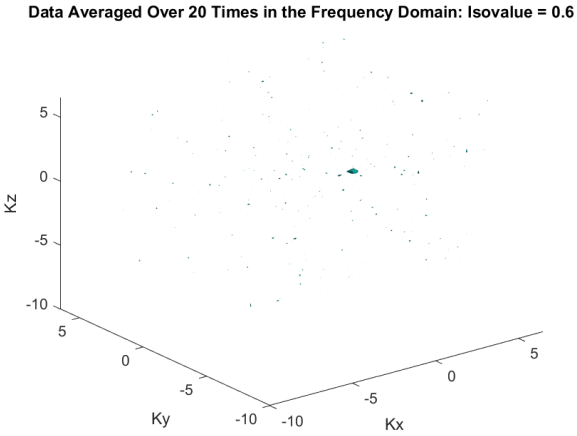
Figure 1 describes raw ultrasound data with white noise. Figure 2 and 3 describes the data averaged over in the frequency domain with different isovalues.



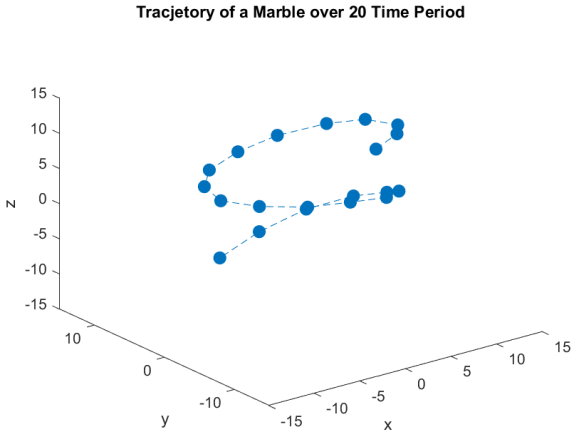
(a) Figure 1



(b) Figure 2



(c) Figure 3



(d) Figure 4

Figure 4 describes the trajectory of the marble after the data has been filtered and converted back into the spatial domain. The final location of the marble should be located at (-5.6250,4.2188 -6.0938).

5 Summary & Conclusion

After transforming the data into the frequency domain using Fast Fourier Transform from the spatial domain and averaging the data over 20 period of time, a central frequency of the marble was located. A 3-D Gaussian filter is then applied to the central frequency to denoise further. The data is then converted back into the spatial domain and a trajectory of the marble is then plotted and shown. The final location of the marble is then marked and eventually will be destroyed with an acoustic wave.

6 Appendix A.

linspace(x1,x2,n): generates n points between x1 and x2 where the spacing is (x2-x1)/(n-1)

fftshift(x): rearranges a Fourier transform x by shifting the zero to the center of the array

isosurface(x,y,z,v): computes and connects point x,y,z with specified isovolume v, simlar to contour lines

fftn(x): returns the multidimensional Fourier transform of a N-D array using Fast Fourier Transform. Th N-D array is the same as using Fast Fourier Transform along each dimension of x.

reshape(A,n,n,n) = reshapes A into a $n \times n \times n$ array.

find(A == B) = returns non-zero indices where A == B

ind2sub(A,B) returns arrays of size A containing the multidimensional subscripts corresponding to indices of B

max(A) = find maximum value of A

plot3(X,Y,Z) = plots coordinates in 3D space

7 Appendix B.

reshape(x): reshape vector x into a matrix of n dimensions where n is 3 in this case

```
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); %shifted

%%
%plotting
%data at time t = 10
for j= 10
    Un(:,:,:)=reshape(Undata(j,:),n,n,n);
    close all, isosurface(X,Y,Z,abs(Un),0.4)
    title("Raw Ultrasound Data");
    xlabel("x");ylabel("y");zlabel("z");
    axis([-20 20 -20 20 -20 20]), grid on, drawnow
    pause(0.1)
end

%%
%part 1
%averaging of spectrum to determine  center frequency
    generated by marble
figure(1)
ave = zeros(64,64,64);
for i = 1:20
     Un(:,:,:)=reshape(Undata(i,:),n,n,n);
     utn = fftn(Un);
     ave = ave + utn;
end

%plots, figure 1-3 corresponds to different isovalues
```

```
figure(1)
ave = abs(fftshift(ave))/20;
maxAve = max(max(max(abs(ave))));
close all, isosurface(Kx,Ky,Kz,ave/maxAve,0.4)
title("Data Averaged Over 20 Times in the Frequency
    Domain: Isovalue = 0.4")
figure(2)
isosurface(Kx,Ky,Kz,ave/maxAve,0.5)
title("Data Averaged Over 20 Times in the Frequency
    Domain: Isovalue = 0.5")
figure(3)
isosurface(Kx,Ky,Kz,ave/maxAve,0.6)
title("Data Averaged Over 20 Times in the Frequency
    Domain: Isovalue = 0.6");
clear xlabel; clear ylabel; clear zlabel;
xlabel("Kx");ylabel("Ky");zlabel("Kz");


%note particular big spot in the back -> center
    frequency of marble

%Finding Coordinates of Maximum Signal
[xs,ys,zs] = ind2sub([n,n,n], find(ave == maxAve));
maxK = [Kx(xs,ys,zs), Ky(xs,ys,zs), Kz(xs,ys,zs)]; %
    coordinates of max signal
%%
%part 2
%filtering data around center frequency to denoise
    using Gaussian filter
%and get path
[kx,ky,kz] = meshgrid(k,k,k); %unshifted for plotting
marble = zeros(20,3); %create array of zeroes to store
    marble coordinate at differnent time periods
tau = 0.4;
k1 = maxK(1);
k2 = maxK(2);
k3 = maxK(3);
filter = exp(-tau*((kx - k1).^2+(ky-k2).^2+(kz-k3).^2))
    ; %define gaussian filter function

for m = 1:20
```

```
69      Un(:,:,:)=reshape(Undata(m,:),n,n,n); %3d matrix of
            raw data
70      utn = fftn(Un); %fft of raw data
71      unft = utn.*filter; %apply filter to transformed
            data
72      unf = (ifftn(unft)); %return filtered data to
            spatial domain
73      maxunf = max(max(max(abs(unf)))); %max of
            filtered data of each time
74      [xm,ym,zm] = ind2sub([n,n,n], find(abs(unf) ==
            maxunf)); %coordinates of marble center at
            differnet times
75      marble(m,:) = [X(xm,ym,zm), Y(xm,ym,zm), Z(xm,ym,zm
            )];
76 end
77
78 %plotting marble trajectory
79 figure(4)
80 plot3(marble(:,1), marble(:,2), marble(:,3), '.--','
       MarkerSize',30)
81 xlim([-L,L]);ylim([-L,L]);zlim([-L,L]);
82 title("Tracjetory of a Marble over 20 Time Period");
83 xlabel("x");ylabel("y");zlabel("z");
84
85 %%
86 %part3
87 acoustic = marble(end,:) %coresponds to -5.625, 4.2118,
       -6.0938
```