Applications of FFT to Filtering Noise from Ultrasound Data

Jeffrey M Abraham

ABSTRACT

The purpose of this report is to show the process of developing an algorithm to filter ultrasound data and find the trajectory of an object through multiple snapshots. This is accomplished using Fourier transforms, averaging, and Gaussian filters.

Keywords: Fast Fourier Transform, Ultrasound, Filtering

I INTRODUCTION AND OVERVIEW

This analysis is being done to track the trajectory of a marble through the intestines of my dog, Fluffy, throughout time from noisy ultrasound data. The location of the marble at the final time step will be used for the focusing of an intense acoustic wave to break up the marble. This is a hypothetical problem posed to develop a method for filtering about an unknown frequency. An example of the unprocessed data can be seen in Figure 1. This analysis functions as a project for an applied mathematics course on computational methods for data analysis, and all data was provided by the instructor.

II THEORETICAL BACKGROUND

Fourier Transforms

The Fourier transform is a process used to break down a periodic function down into a combination of *sin* and *cos* functions of different frequencies.
The Fourier transform can be defined in multiple ways, but for the purposes of this analysis the transform(1) and inverse transform(2) are defined below.

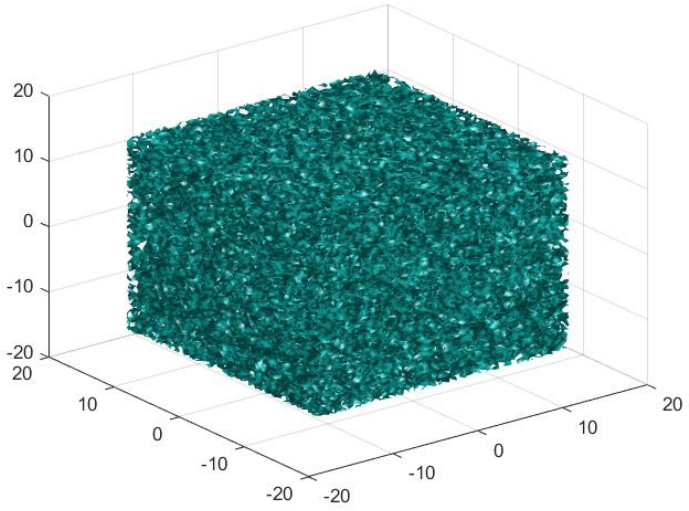

Figure 1. Unprocessed data plotted using the isosurface command

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

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

For this analysis however we need to use a discrete Fourier series. Matlab defines its FFT discrete analysis as:

$$\tilde{f}(k) = \sum_{x=1}^{n} f(x) e^{-\frac{2\pi i(x-1)(k-1)}{n}} \tag{3}$$

$$f(x) = \sum_{k=1}^{n} \tilde{f}(x) e^{\frac{2\pi i(x-1)(k-1)}{n}} \tag{4}$$

Averaging

Averaging is a method of removing noise by averaging multiple time steps of data in the frequency basis. This involves Fourier transforming and adding together frequency amplitudes. You lose all location data when averaging, so averaging is most useful for finding frequency signatures.

Gaussian Filtering

Gaussian filtering is the process of multiplying the amplitudes by a Gaussian function centered around a signature frequency to reduce all noise far from that frequency. Other functions can be used for filtering, but a Gaussian function will be used for this analysis.

$$f(x) = e^{-a(x-x_0)^2} \tag{5}$$

$$f(x)_{4D} = e^{-a[(x-x_0)^2+(y-y_0)^2+(z-z_0)^2]} \tag{6}$$

The form of a 2D Gaussian function is shown in (5) and extended to a 4D Gaussian functions as would be necessary to filter the ultrasound data is shown in (6).

III ALGORITHM IMPLEMENTATION AND DEVELOPMENT

In order to do FFT, first I needed to create meshgrid lists for X,Y,Z in position space as well as Kx,Ky,Kz in frequency space. This can be seen in lines (4-14) in Appendix B(later referred to as (B 4-14)). For frequency space, matlab assumes a 2π periodic signals for FFT, so the frequency arrays had to be scaled by $\frac{2\pi}{2*L}$. In addition they needed to be shifted to have the 0 frequency entry in the center.

In order to filter the noise from the ultrasound data first I needed to determine the frequency signature of the marble. This was done by averaging in the frequency basis (B 24-31). The first line of the for loop here is used to pick data at each successive time step. The frequency is then found by taking the index of the maximum amplitude of the averaged values, and finding the frequencies at those points in the x, y, and z frequency meshgrids(B 32-36).

With the center frequency determined, the Gaussian filter could be defined, and the data could be filtered in a for loop, normalized, the marble center (found by recording the index of the maximum amplitude in position space) could be recorded, and the isosurface plotted. Maximum was used to find the center because the maximum intensity in ultrasound data represents the point where highest percentage of the signal reflected from. So for determining where to focus an acoustic wave to break up the marble, this point would be the most solid point on the marble, and thus a good target. Through testing, this method also appeared to give better center locations than averaging all of the vertices for each time step.

From there tuning of the isovalue and 'a' value in the filter was done to achieve less noise and more accurate graphing.

Besides specific value tuning, this algorithm could be used to find any object in noisy data.

IV COMPUTATIONAL RESULTS

The central frequency was determined to be at $[Kx, Ky, Kz] = [-4.8171, 5.6549, -6.7021]$
After tuning, an isovalue of 0.9 and an 'a'(6) of 5.5 achieved the clearest results.

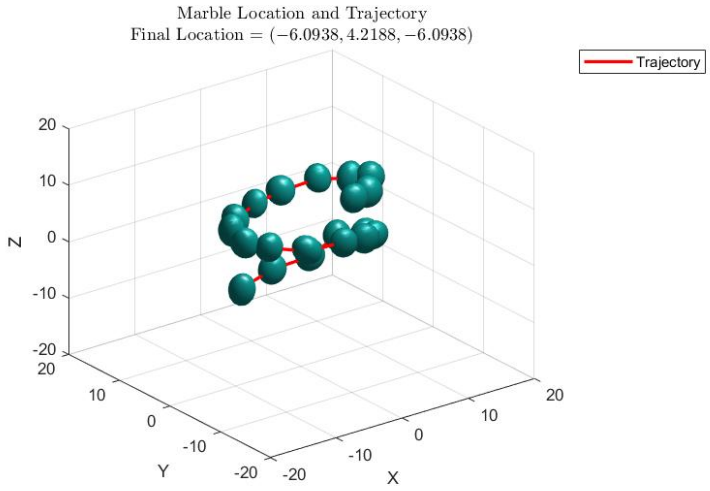

Figure 2. Unprocessed data plotted using the isosurface command

The trajectory of the marble can be seen in Figure 2. The marble starts at the top of the loop and moves in a spiral in the $-z$ direction. The final location of the center of the marble was $(x,y,z) = (-6.0938, 4.2188, -6.0938)$

V SUMMARY AND CONCLUSIONS

Through this exercises I developed an algorithm for filtering noisy signals over time, that could easily be adapted to work with signals of different dimensions. The entire code excluding the example plot runs in 3.1 seconds. This quick run time allows for the doctors to break up the marble in Fluffy's intestines without suffering.

The fast Fourier transform is a quick and effective tool that when combined with a Gaussian filter can easily de-noise a signal and find the location of known frequencies. Averaging can also be used to de-noise data in the frequency basis, but as a result you lose the location data. When combining both methods, you can find a target frequency within a noisy signal and filter noise around that frequency to get location data.

APPENDIX A.

Matlab Functions

fftn(X)
n^{th} dimensional fast Fourier transform.

fftshift(X)
Shifts frequencies to have zero frequencies centered

ifftn(Y)
n^{th} dimensional inverse fast Fourier transform.

isosurface(X,Y,Z,V,isovalue)
Plots vertices at points that correspond to entries of V with the isovalue and connects them with faces. Essentially a contour plot for 4 dimensions

legend(subset,'Label')
creates a legend for only the plots contained in subset

[M,I] = max(A,[],'all','linear')
returns maximum entry as M and the linear index of that value in I

reshape(A,sz)
Takes a linear data-set and reshapes it into the shape defined with sz.

APPENDIX B.

Matlab Code

```
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);
%%
%Example Plot of Original Data
Ex(:,:,:)=reshape(Undata(1,:),n,n,n);
figure(1)
isosurface(X,Y,Z,abs(Ex),0.4)
axis([-20 20 -20 20 -20 20]), grid on, drawnow
xlabel('X');ylabel('Y');zlabel('Z');
hold on;
%%
Ua = zeros(n,n,n);
an = 20;
for j=1:an
    Un(:,:,:)=reshape(Undata(j,:),n,n,n);
    %Average Finding
    Ua = Ua + fftn(Un);
end
Ua = abs(Ua/an);
%Finding Frequency Signature
[M,I] = max(Ua,[],'all','linear');
kx0 = Kx(I);
ky0 = Ky(I);
kz0 = Kz(I);
%Normalizing and Plotting
Ua = Ua/max(Ua,[],'all');
figure(2)
close all, isosurface(Kx,Ky,Kz,abs(Ua),0.75);
hold on
axis([-5 -4.5 5.5 6 -7 -6.5]), grid on, drawnow

%%
%Building Filter Function
a = 5.5;
f = exp(-a*((Kx-kx0).^2 + (Ky-ky0).^2 + (Kz-kz0).^2));

isv = 0.9;
marble = zeros(20,3);
for j=1:20
    Un(:,:,:)=reshape(Undata(j,:),n,n,n);
```

```
53      %Applying Filter
54      unft = f.*fftn(Un);
55      unf = abs(ifftn(unft));
56      unf = unf/max(unf,[],'all');
57      [M2, I2] = max(unf,[],'all','linear');
58      %Plotting Filtered Data
59      marble(j,:) = [X(I2) Y(I2) Z(I2)];
60      figure(3)
61      isosurface(X,Y,Z,abs(unf),isv), grid on, drawnow
62      axis([-20 20 -20 20 -20 20]);
63      hold on
64  end
65
66  %plotting Trajectory
67
68  plot3(marble(:,1),marble(:,2),marble(:,3),'r','Linewidth',2)
69  grid on
70  fp = marble(20,:);
```