Homework 1: An Ultrasound Problem

Ozan Erdal

January 24$^{\text{th}}$, 2020

Abstract: The objective of this problem was to become familiar with one format in which ultrasound data can be presented and the shortcomings associated with such data. Using ultrasound, a veterinarian obtained information on the spatial variations of a subsection of a dog's intestines in search of a swallowed marble. Due to the movement of the animal, the ultrasound data was noisy. With the help of techniques like the fast Fourier transform, averaging, and filtering, the noise was removed and the marble was made visible for the veterinarian to destroy using a focused acoustic wave.

I. Introduction and Overview

The premise behind this lab was to examine one of the many ways in which noise can be introduced to data during the collection process. By sampling noisy data over a period of time, the noise can then be removed or reduced using signal processing techniques. Since the data is representative of the motion of an object through space over time, the collected data can also be plotted to reveal a trajectory of the object. By using MATLAB to process and plot the data, a clear path can be determined with relatively high accuracy.

II. Theoretical Background

The Fourier transform is a powerful tool for representing a function as a combination of cosines and sines. The transform is defined on the domain $x \in [-\infty, \infty]$ and is given by

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

The Fourier transform additionally has an inverse given by

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

Unfortunately, for use with computational software, an interval of infinite length is not feasible. As a result of this limitation, a similar technique for performing an integral transformation over a finite interval, the fast Fourier transform (FFT), was developed and will be used extensively to determine several important quantities going forward. The FFT algorithm is much faster asymptotically than the standard Fourier transform, so for usage with large volumes of data, it provides excellent accuracy with a significantly lower operation count.

The FFT also has an inverse and has been expanded to usage in n-dimensions. MATLAB provides standard support for all of these variations, but since the question relates to ultrasound data, the 3-dimensional fast Fourier transform was used, namely utilizing the *fftn()*, *ifftn()*, and *fftshift()* MATLAB functions, which are discussed further in **Appendix A**.

The FFT procedure provides information for time-frequency analysis of the transformed data. In this case the transformation was used to find the *central frequency*, the frequency which occurs most frequently in the signal. After determining this value for each spatial dimension through an averaging procedure, a filter can applied at each time step to significantly remove noise. The averaging and filtering process are discussed at length in **Section III**.

The averaging process relies on the fact that the disturbance in the data is assumed to be

white noise which has a key property in that it is *zero mean*. Averaging several collections of white noise will remove any peaks and produce a uniform distribution. By using averaging to denoise the data, the white noise will become uniformly distributed across the signal, minimizing its effect.

III. Algorithm Implementation and Development

The algorithms used in this procedure were averaging and filtering. They will be discussed and modeled with pseudo-code, and the full code for each procedure will be available in **Appendix B**.

The original data before applying any algorithms is shown in **Figure 1** for comparison with the Fourier transform.

The data was loaded in and the computational and Fourier domains were set. The data had 64^3 columns, so the number of Fourier modes (n) was set to 64. The Fourier domain was thus broken down into 64 wave numbers for Fourier analysis. The spatial domain was defined to be of length L, so the the x, y, and z vectors ranged from $[-\frac{L}{2}, \frac{L}{2})$ with n values spaced $\frac{1}{n+1}$ apart and with the upper bound excluded. The vector of Fourier modes was constructed in a particular way for visualization. The points were constructed from `[0:(n/2-1) -n/2:-1]` and then scaled by a factor of $\frac{2\pi}{2L}$ since the interval was not a standard $[-\pi, \pi]$ interval on which sines and cosines are periodic. This vector is then shifted to be strictly increasing for plotting purposes using the *fftshift()* function.

III.A. Averaging

Data: ultrasound data
Result: averaged data as well as a frequency signature.
initialize cumulative sum of size nxnxn;
for *each time step* **do**
    reshape the data to nxnxn matrix;
    3-dimensional *fft()* on matrix;
    add result to cumulative sum;
end
divide the cumulative sum by the number of time steps;

Algorithm 1: Averaging process

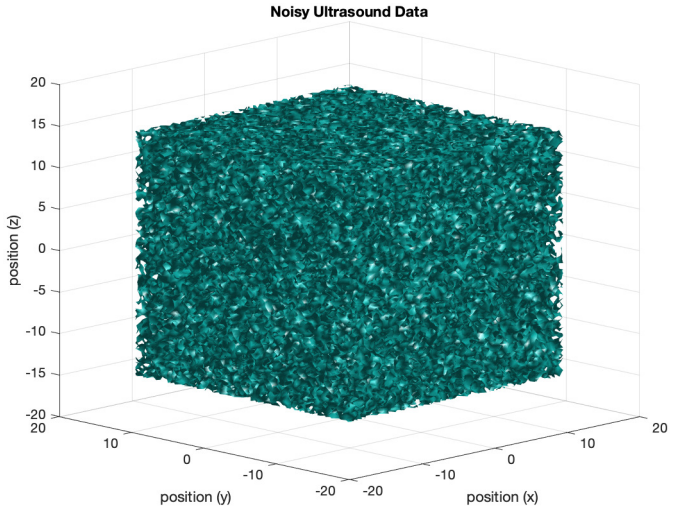

Fig. 1: Noisy Data with Marble Hidden

The end result of the algorithm was denoised frequency information. From this point, the *isosurface()* MATLAB command was used to determine the frequency signature. By setting the *isovalue* to just below the overall maximum value of the averaged matrix, a small area of constant value appeared in the image. This region is shown in **Figure 2** with a data tip on the area of interest, showing the *kx*, *ky*, and *kz* values representative of the frequency signature.

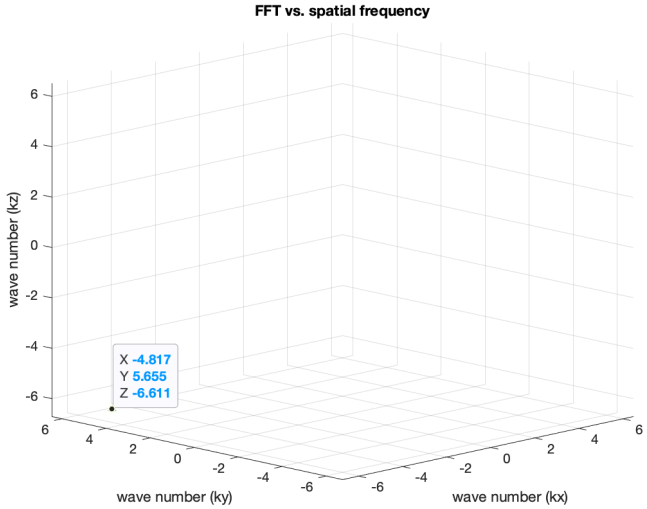

Fig. 2: Frequency signature of averaged FFT

III.B. Filtering

After determining the frequency signature, the filtering function was applied to the noisy data in the Fourier domain with the k_0 value for each dimension. The *tau* value was adjusted until a round clear object was very clearly visible. The final result of this filtering process, the marble swallowed by the dog is shown in **Figure 3**.

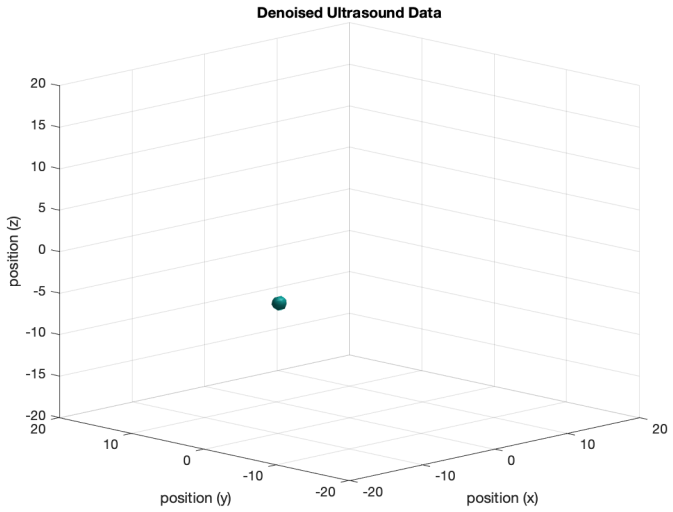

Fig. 3: Marble Location at Final Time Step

By applying this filter to data from the final row of the data, the end position of the marble was determined so that the acoustic pulse could be delivered precisely. Additionally, by applying this filter at each time step and recording the position of the marble, 3 vectors each of length equal to the number of time steps was constructed containing the x, y, and z coordinates of the marble. Using the *plot3()* function, the trajectory formed by the marble as it moved through the intestines of the dog was plotted and is shown in **Figure 4**.

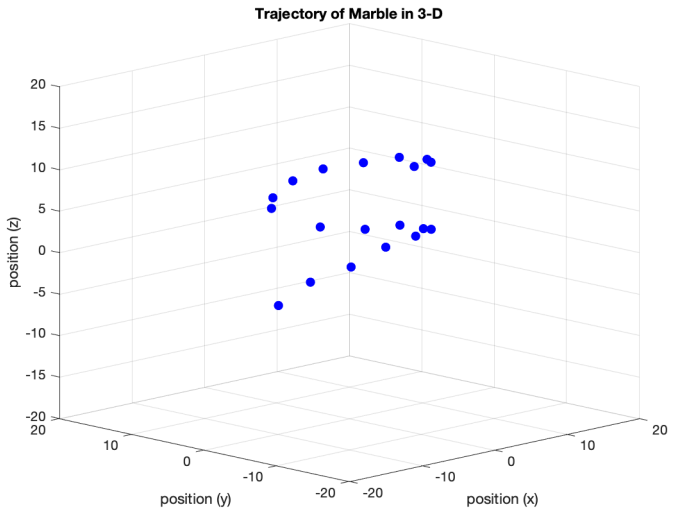

Fig. 4: Marble Trajectory

IV. Computational Results

After the averaging procedure, the values of the frequency signature were determined to be:

$$k_{0x} = -4.817, k_{0y} = 5.655, k_{0z} = -6.611$$

A filter precision of $\tau = 0.5$ was used in the isosurface plot in Figure 3 along with the k_0 values specified. The vectors of trajectory coordinates was recorded by hand and is therefore hard-coded and viewable in **Appendix B**. The end position of the marble is a particularly important value however, so it is worth mentioning. At the final time step, the coordinates of the center of the marble were:

$$x_{final} = 4.688, y_{final} = -4.29, z_{final} = 10.31$$

V. Summary and Conclusions

In conclusion, averaging can be used in the Fourier domain to reduce the effect of white noise. From this a central frequency, or *frequency signature* can be identified by visualizing the averaged data. A filter, in this case, a Gaussian filter, can be applied in the Fourier domain to remove or keep specific frequencies. Inverse Fourier transforming the filtered data back into the spatial domain provides a clear image if done properly. The marble was very clearly detectable despite the original image looking quite different.

VI. Appendix A - MATLAB functions

reshape(U,n,n,n)	Reshapes data U to dimension nxnxn.
fftn(U)	n-dimensional fast Fourier transform. U is the the data to transform.
ifftn(Ut)	n-dimensional inverse fast Fourier transform. Ut is data to transform.
fftshift(Ut)	Rearranges Fourier transform values. Moves 0 frequency component to center.
meshgrid([x,y,z])	Converts vectors to matrices. Results in size n^3 matrix.
isosurface(X,Y,Z,V,isoval)	Plots isovolume surfaces. X, Y, and Z are axes processed through meshgrid.
plot3(x,y,z)	Plots x, y, and z vectors as a point at each index.

VII. Appendix B - MATLAB code

Ultrasound Denoising

```
clear; close all; clc;

%% Starter Code
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);

for j=1:20
    Un(:,:,:)=reshape(Undata(j,:),n,n,n);
    close all, isosurface(X,Y,Z,abs(Un),0.4)
    axis([-20 20 -20 20 -20 20]), grid on, drawnow
    pause(1)
```

```
end

%% Figure 1
close all

figure
isosurface(X,Y,Z,abs(Un),0.4)
title('Noisy Ultrasound Data')
xlabel('position (x)')
ylabel('position (y)')
zlabel('position (z)')
gca.FontSize = 14;
view(-45,15)
axis([-20 20 -20 20 -20 20]), grid on, drawnow
print -depsc original_data.eps

%% Part 1: Determine the Frequency Signature
close all
s=20;
Utnave=zeros(n,n,n);
for j=1:20
    Un(:,:,:)=reshape(Undata(j,:),n,n,n);
    Utn=fftn(Un);
    Utnave=Utnave+Utn;
end
Utnave=fftshift(Utnave)/s;

%% Figure 2
close all

figure
isosurface(Kx,Ky,Kz,abs(fftshift(Utnave)),max(abs(Utnave(:)))-50)
title('FFT vs. spatial frequency')
xlabel('wave number (kx)')
ylabel('wave number (ky)')
zlabel('wave number (kz)')
```

```
gca.FontSize = 14;
view(-45,15)
axis([min(Kx(:)) max(Kx(:)) min(Ky(:)) max(Ky(:)) min(Kz(:)) max(
    Kz(:))]), grid on, drawnow
% print -depsc frequency_signature.eps

%% Part 2: Filter the Data
close all;

kx=-4.817;ky=5.655;kz=-6.611;
tau=0.5;

filter1d=exp(-tau*(k-kx).^2); % Define the filter
filter=((tau/2/pi)^(3/2))*exp(-tau*(((Kx-kx).^2)+((Ky-ky).^2)+((Kz
    -kz).^2))/2); % Define the filter

for j=1:s
    Un(:,:,:)=reshape(Undata(j,:),n,n,n);
    Utn=fftn(Un);
    Utnf=filter.*Utn; % Apply the filter to the signal in
        frequency space
    Unf=ifftn(Utnf);
    close all
    isosurface(X,Y,Z,abs(Unf),0.9*max(abs(Unf(:))))
    axis([-20 20 -20 20 -20 20]), grid on, drawnow
    pause(0.5)
end

%% Figure 3
close all

figure
isosurface(X,Y,Z,abs(Unf),0.9*max(abs(Unf(:))))
title('Denoised Ultrasound Data')
xlabel('position (x)')
ylabel('position (y)')
```

```
88 zlabel('position (z)')
89 gca.FontSize = 14;
90 view(-45,15)
91 axis([-20 20 -20 20 -20 20]), grid on, drawnow
92 print -depsc marble_location.eps
93
94 %% Part 3: Plot the Trajectory
95 close all
96
97 x_traj=[4.688 8.438 10.31 9.042 6.094 1.333 -3.315 -7.6 -9.844
       -7.5 -2.612 2.344 6.562 9.375 9.844 7.969 4.219 -0.9375
       -5.625];
98 y_traj=[-4.29 -2.857 -0.435 2.124 4.131 4.936 4.467 2.932 0.883
       -3.502 -4.804 -4.66 -3.683 -1.942 0.669 2.923 3.957 4.413
       4.1375];
99 z_traj=[10.31 9.844 9.375 9.375 8.906 8.906 8.438 7.5 7.031 5.178
       4.219 3.75 2.344 1.406 0 -1.406 -3.281 -4.219 -6.094];
100
101
102 %% Figure 4
103 figure
104 plot3(x_traj,y_traj,z_traj,'b.', 'MarkerSize', 20)
105 title('Trajectory of Marble in 3-D')
106 xlabel('position (x)')
107 ylabel('position (y)')
108 zlabel('position (z)')
109 gca.FontSize = 14;
110 view(-45,15)
111 axis([-20 20 -20 20 -20 20]), grid on, drawnow
112 print -depsc marble_trajectory.eps
```