Use of Fourier Transform to remove noise from Ultrasound Signals

Connor Schleicher

January 24, 2020

Abstract

Signal processing is an important process in many different applications from radar, to image processing on a cell phone. A major complication in signal processing is denoising the data and removing everything that is not the object in question. One method involves the use of averaging to obtain the central frequency, followed by applying a filter to dampen out the noise in the signal. This analysis starts with raw data from an ultrasound, taken at a constant frequency, and is analyzed to find the frequency and location of a marble as it travels through a dog's intestines. Averaging the noise, and the application of a three dimensional Gaussian filter proved effective to locate the marble as it moved along its path.

1 Introduction and Overview

Digital signal processing is used daily in our modern lives from digital image processing in our phones, radar detection, ultrasounds, and speech processing and cognition in our digital AI based assistants [1]. A subset of signal processing is time frequency analysis which allows us to discern time frequency data from spatial data points. The processes handling these signals are fast and accurate. Without the ability to filter out the noise all these signals being transmitted would be meaningless.

In this analysis a raw digital signal is provided with no information about frequency and with ample noise surrounding the true object reflecting the signal. The object in question is a marble which was accidentally ingested by an unsuspecting dog. In order to begin to find the location of the object, the central frequency must be obtained. The initial signal is provided as spatial data over twenty different samplings. It is possible to average the noise out of the signal by transforming the raw signal and adding it to the previously transformed signal. This is repeated over all the samples, then divided by the total samples to obtain the average. This creates a new signal where all the noise cancels and all that is left is the central frequency [3].

After the central frequency is known, a filter must be applied to each sampling to de-noise the signal. This filter is applied over a narrow enough bandwidth to catch the signal while trying to not be too wide and losing resolution. A simple Gaussian filter is sufficient to capture the true signal and find the location of the signal in the sea of noise.

With the filter applied to the data, the only signal that is left is the true signal. The location data from this signal can be extracted and plotted showing the path of the object. The final location point can be assumed to be the current location of the object at the time of measurement.

2 Theoretical Background

2.1 Fourier Transform

The Fourier transform can translate a signal of time into its frequency components. The Fourier transform is defined by:

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

Along with the corresponding inverse Fourier transform to get back to the original function:

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

Linear computations done in the Fourier space has a corresponding operation in the functional space [2]. The advantage of translating something into Fourier space is there is usually a computational advantage in doing so. This computational advantage is especially noticeable in taking derivatives. It can be proven that the derivative in Fourier space simplifies to this linear relationship [3]:

$$\widehat{f'(x)} = ik\widehat{f(x)} \tag{3}$$

2.2 Fast Fourier Transform

The Fast Fourier Transform (FFT) is a numerical routine used to perform the forward and backward Fourier transforms. The FFT is a discrete Fourier transform assuming periodic boundary conditions on interval $x \in [-L, L]$ [3]. One of the main advantages of the FFT is its speed compared to other similar algorithms. It reduces the order of operations from $2N^2$ to `O(2NlgN)`, where lg is the base-2 logarithm [4].

2.3 Gaussian Filter

The Gaussian, or normal distribution function acts as a low-pass filter since it eliminates high-frequency components over the defined bandwidth [3]. The filter in three dimensions can be defined by the following:

$$f(kx, ky, kz) = e^{-\tau((kx-fx)^2+(ky-fy)^2+(kz-fz)^2)} \tag{4}$$

The bandwidth is defined by τ and is used to narrow the frequency window to provide more or less definition from the filter.

3 Algorithm Implementation and Development

In solving this question the program was broken into three distinct sections, separating the logical steps in solving the problem:

1. Initialization of program (Algorithm 1)
2. Average the noisy signal and find the central frequency (Algorithm 2)
3. Create a filter and plot the marble's path (Algorithm 3)

Algorithm 1: Initialization

```
Import data from Testdata.mat
Initialize variables and create coordinates for the Fourier modes on a periodic domain
ISOSURFACE(X,Y,Z,abs(Un),0.4)
```

Algorithm 2: Averaging Signal and Finding Central Frequency

```
Initialize U average to zeros matrix
for $j = 1 : 20$ do
    Extract measurement $j$ from Undata
    Add the Fourier transform of Undata to U average
end for
Shift and take absolute value of U average
Get max value from U average, this is the central frequency
```

Algorithm 3: Averaging Signal and Finding Central Frequency

```
Create 3D Gaussian filter around central frequency location
Initialize variables to zeros vectors
for $j = 1 : 20$ do
    Extract measurement $j$ from Undata
    Apply Fourier transform and shift Undata
    Multiply the transformed and shifted data by the filter
    Get max value of the filter and store the matrix index
    Find and store X,Y,Z locations from the matrix index
end for
Plot marble path by plotting X,Y,Z locations
```

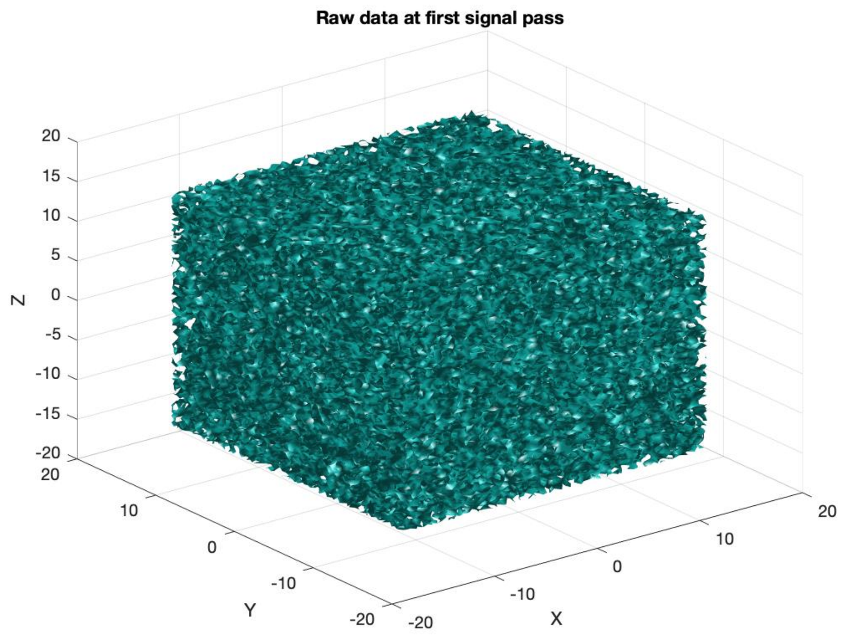

Figure 1: Isosurface plot of raw data for first signal scan.

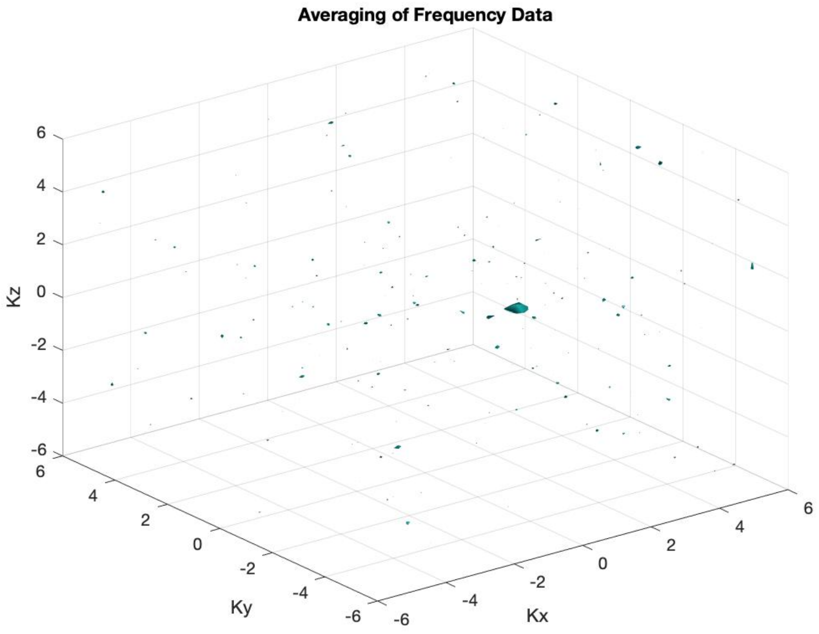

Figure 2: Central frequency persists after averaging noise out of the signal.

4 Computational Results

After importing the data, the raw data was plotted using the isosurface function in Matlab, see Figure 1. From this image there is no visually identifiable signal.

The method of averaging was able to de-noise the data sufficiently to identify the central frequency. This process, defined in Algorithm 2, essentially filtered out the excess frequencies and the resulting isosurface plot showed a clear signal that persisted throughout the averaging process. The larger grouping of signals seen in figure 2 shows the signal that persisted throughout the averaging.

The max value of the average signal was `5,437`, with an index location of `Row 28, Column 42, Slice 33`. With the central frequency known, the Gaussian filter could be applied to each signal record. By finding the max frequency for each record the marble's location could be found and plotted, as seen in Figure 3. The final position of the marble can be seen in red in Figure 3, and is at position `(X: -5.6250, Y: 4.2188, Z: -6.0938)`.

5 Summary and Conclusions

Signal processing is a process occuring in our everyday life. The ability to quickly filter out the important information and remove the noise from the raw signal is a vital process. In this analysis an ultrasound signal was processed to identify the underlying frequency which it was emitted at. Once the frequency was determined, the noise could be removed and the location of the marble was determined. This location data can provide an accurate path of the marble as it moves through the dog.

References

[1] *Digital Signal Processing.* URL: `en.wikipedia.org/wiki/Digital_signal_processing`.

[2] *Fourier Transform.* URL: `https://en.wikipedia.org/wiki/Fourier_transform`.

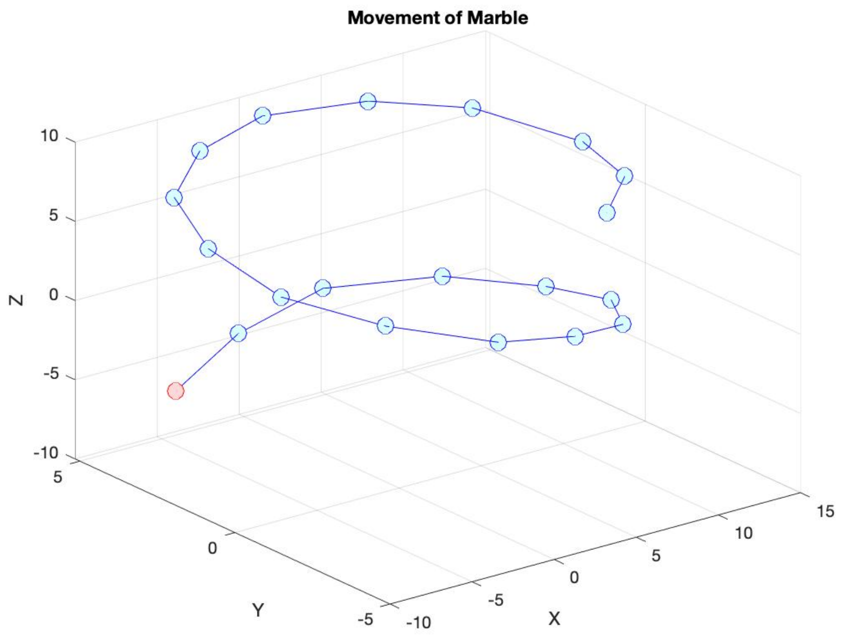

Figure 3: Marble path found starting from the top and spiraling downward.

[3] Jose Nathan Kutz. *Data-driven modeling & scientific computation: methods for complex systems & big data*. Oxford University Press, 2013.

[4] Eric W. Weisstein. *Fast Fourier Transform*. URL: `http://mathworld.wolfram.com/FastFourierTransform.html`.

Appendix A MATLAB Functions

Add your important MATLAB functions here with a brief implementation explanation. This is how to make an **unordered** list:

- `isosurface(X,Y,Z,abs(Un),0.4)` constructs an isosurface plot along the X,Y,Z axes. The data, `abs(Un)` acts as the volume data being displayed. The isosurface value, `0.4`, specifies where the isosurface data is computed.
- `fftn(Un)` creates the `N` dimensional Fourier Transform of `Un` using the fast Fourier transform algorithm. The output is the same size of as the input function and assumed the same dimension of the input function.
- `fftshift(Uave)` rearranges Fourier transformed data by shifting the zero-frequency component to the center. This is done by alternating half space intervals over the number of dimensions in the input argument.
- `reshape(Undata(1,:),n,n,n)` converts the vector `Undata` into a `NxNxN` matrix.
- `ind2sub(size(Uave),idx)` translates the index of a value `idx` to the row, column, slice's of the matrix

Appendix B MATLAB Code

Add your MATLAB code here. This section will not be included in your page limit of six pages.

```
%Connor Schleicher AMATH 582 HW 1

%% Initialize the program
% initial code from HW prompt for initializing 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);

% initial figure to show the unusefulness of the noisy signal
figure(1)
Un(:,:,:)=reshape(Undata(1,:),n,n,n);
close all, isosurface(X,Y,Z,abs(Un),0.4)
axis([-20 20 -20 20 -20 20]), grid on, drawnow
xlabel('X'); ylabel('Y'); zlabel('Z');
title('Raw data at first signal pass')

%% Average the noisy singal and find central frequency
% iterate through the first 20 realizations
% averaging the signal to find the central frequency
% iso surface to create visualization to see the averaging happen

Uave = zeros(n,n,n); % initialize the averaged variable
for j=1:20
    figure(2);
    Un(:,:,:)=reshape(Undata(j,:),n,n,n);
    Uave = Uave + fftn(Un); % averaging the shifted transform of the data
    isosurface(Kx,Ky,Kz,abs(fftshift(Uave))/max(abs(fftshift(Uave(:)))),0.6)
    axis([-6 6 -6 6 -6 6]), grid on, drawnow
    xlabel('Kx'); ylabel('Ky'); zlabel('Kz');
    title('Averaging of Frequency Data')
    pause(1)
    if j < 20 % closes figure for next iteration isosurface won't erase old data
        close(2);
    end
end

Uave = abs(fftshift(Uave)); %shift and take abs value of averaged data

% find the maximum value of the averaged signal
[value,idx] = max(Uave(:));
[r,c,p] = ind2sub(size(Uave),idx);

% sanity check of max value calculation
fprintf('Max value of function (abs): %d \n',abs(value));
fprintf('Uave value at position r,c,p (abs): %d \n', abs(Uave(r,c,p)));

% print locaton of central frequency
fprintf('Location of central frequency: %d, %d, %d \n', r,c,p);
```

```
%% Create Filter and Plot Marble Path
% create the filter to de-noise the data around the central frequency
tau = 0.5; % bandwidth of filter
% a simple gaussian filter across the three spatial dimensions
filter = exp(-tau*((Kx - ks(c)).^2 + (Ky - ks(r)).^2 + (Kz - ks(p)).^2));

% initialize some vectors to store values from each signal pass
value2 = zeros(1,20);
idx2 = zeros(1,20);
r2 = zeros(1,20);
c2 = zeros(1,20);
p2 = zeros(1,20);
locations = zeros(20,3);

% Apply the filter to each signal (loop of 20 iterations)
for i = 1:20
    Un(:,:,:)=reshape(Undata(i,:),n,n,n);

    Unt = fftshift(fftn(Un)); % applying fft and shift to the raw data

    Utnf = Unt.*filter; % apply the Gaussian filter to each element of the transformed dat

    U = ifftn(Utnf); % inverse transform to get back to the original signal

    % getting value and index position of the max value
    [value2(i), idx2(i)] = max(U(:));

    % translating the index to the row, column, slice's of the matrix
    [r2(i),c2(i),p2(i)] = ind2sub(size(U),idx2(i));
    locations(i,:) = [X(r2(i),c2(i),p2(i)), Y(r2(i),c2(i),p2(i)),...
        Z(r2(i),c2(i),p2(i))]; % sets XYZ coordinates in a matrix to be used for plotting
end

figure (3)
% Plotting the position of the marble for each singal recording
% Marble is starting at the top (highest Z position) and working downwards
plot3(locations(:,1),locations(:,2),locations(:,3),...
    '-o','Color','b','MarkerSize',10,'MarkerFaceColor','#D9FFFF')
hold on

% Replotting the last point of the marble to color it red to stand out
plot3(locations(end,1),locations(end,2),locations(end,3),...
    '-o','Color','r','MarkerSize',10,'MarkerFaceColor','#FFD9D9')
xlabel('X'); ylabel('Y'); zlabel('Z');
title('Movement␣of␣Marble');
grid on;

fprintf('The␣final␣location␣of␣the␣marble␣is␣(X,Y,Z):␣%0.4f,␣%0.4f,␣%0.4f␣\n' ...
    ,locations(end,1),locations(end,2),locations(end,3))
```