Computational Analysis and Processing of Three-Dimensional Ultrasound Data

Maxwell Weil

Abstract

In this homework, we will be investigating methods for analyzing noisy datasets. To achieve this, we will use MATLAB software to explore and ultimately make use of noisy 3D ultrasound data. However, before this is done, it is imperative that the basic principles of both the Fourier transform and filtering techniques are well understood. An overview of these concepts and the theoretical ideas backing them will be reviewed, along with how they have been implemented computationally in this work.

I. Introduction

Within the field of engineering and science, there are many complex problems that require analysis of signals. Signals can come in all forms: soundwaves, light, heat, even gravity. Having the ability to break signals down into their fundamental elements is an essential tool for understanding how they work. Perhaps one of the most useful tools for accomplishing this, is the Fourier transform. The Fourier transform is often used to transform a signal in the time or spatial domain into the frequency domain, where different aspects of a signal can be examined.

In this work, a combination of the Fourier transform and filtering techniques will be used to analyze sample ultrasound data using MATLAB software. This data contains an object moving throughout space over a period of time. The goal is to locate this object, visualize it, and track its motion. In doing this, signal analysis techniques will be applied, exploring how these algorithms can be used to solve a practical issue. Prior to covering these algorithms, a solid understanding of the basis of signal processing will need to be established. Following this, the methods and results from this work will be discussed and summarized, along with the detailed equations and code used.

II. Theoretical Background

The Fourier transform plays an integral role throughout this work, so understanding it is essential. As mentioned previously, the basis of the Fourier transform is the Fourier Series, shown in equation 1. All equations have been adapted from [1].

$$f(x) = \frac{a_0}{2} + \sum_{n=1}^{\infty}(a_n \cos \mathrm{nx} + b_n \sin \mathrm{nx}) \quad x \in (-\pi, \pi] \qquad \text{EQ. 1}$$

In this formula, f(x) represents a continuous function in the range (-π, π]. The RHS shows the Fourier series, where the given function is broken down into an infinite sum of sines and cosines with varying frequencies, given by the values of n. While an infinite number of sines and cosines are summed, the actual frequencies found using this series are discrete and noninfinite. The coefficients, given by a_n and b_n, can be defined using equations 2 and 3.

$$a_n = \frac{1}{\pi}\int_{-\pi}^{\pi} f(x)\cos(nx)\,dx \qquad \text{EQ. 2}$$

$$b_n = \frac{1}{\pi}\int_{-\pi}^{\pi} f(x)\sin(nx)\,dx \qquad \text{EQ. 3}$$

In short, these equations show that any f(x) within interval (-π, π] can be recreated using only sine and cosines, even nonperiodic functions. This is because this equation assumes that f(x) is periodic, with a period of 2π. Note that this assumed period can actually be changed to any value, L, by recalling the that frequency is equal to 2π divided by the period. What happens when this period grows infinitely large, so that the interval of f(x) is now (-∞, ∞)? This is where the Fourier transform, shown in equation 4, comes from. Recall that Euler's formula allows for the transformation of the cosine and sine into a single exponential (allowing for a single integral with imaginary components).

$$F(k) = \frac{1}{\sqrt{2\pi}} \int_{-\infty}^{\infty} e^{-ikx} f(x) dx \quad \text{EQ. 4}$$

The Fourier transform differs from the Fourier series because it does not have to assume f(x) is periodic on any bounded interval. This produces a continuous range of frequencies, each given by k. This allows for the transformation of a function in the domain of x (usually time or space) into the domain of k. This equation can also be inverted, in order to transform a function in the frequency domain into the time or space domain. The inverse Fourier transform is given in equation 5.

$$f(x) = \frac{1}{\sqrt{2\pi}} \int_{-\infty}^{\infty} e^{-ikx} F(k) dk \quad \text{EQ. 5}$$

Thus far, all of these equations have assumed that the function f(x) is continuous. But most practical data is discrete. This is because, even when a signal is continuous, it must be sampled in order to be recorded. Accordingly, the rate at which a signal is sampled gives the sampling frequency (Fs), which will come into play later. If f(x) is discrete, the discrete Fourier transform and its inverse must be used, as given in equations 6 and 7.

$$F_\omega = \frac{1}{\sqrt{\pi}} \sum_{n=0}^{N-1} f_n e^{\frac{-i\omega n}{N}} \quad \text{EQ. 6}$$

$$f_n = \frac{1}{N\sqrt{\pi}} \sum_{k=0}^{N} F_\omega e^{\frac{-i\omega n}{N}} \quad \text{EQ. 7}$$

In this formula, N represents the total number of samples taken, n is the current sample, and ω is the frequency bin (which is limited due to the sampling frequency and number of samples). Essentially, n represents a discrete x, and ω/N represents a discrete k in our previous equations. Using this transform, a discretized signal can be transformed into the frequency components that compose it. However, the sampling size plays an important role in what frequencies can make up a signal. High frequency sinusoids cannot be effectively captured by a low sampling frequency, which can result in aliasing. Figure 1 illustrates this concept in detail. Because of this, only frequencies lower than Fs/2 should be considered to contribute to the signal.

In this work, the discrete Fourier transform, and its inverse, are used to analyze ultrasound data. However, a special algorithm known as a fast Fourier transform, or FFT, will be used to compute these. The FFT algorithm is more efficient than the traditional discrete Fourier transform, though the precise explanation is outside of the scope of this paper. For all intents and purposes, the FFTs used throughout this project function as discrete Fourier transforms.

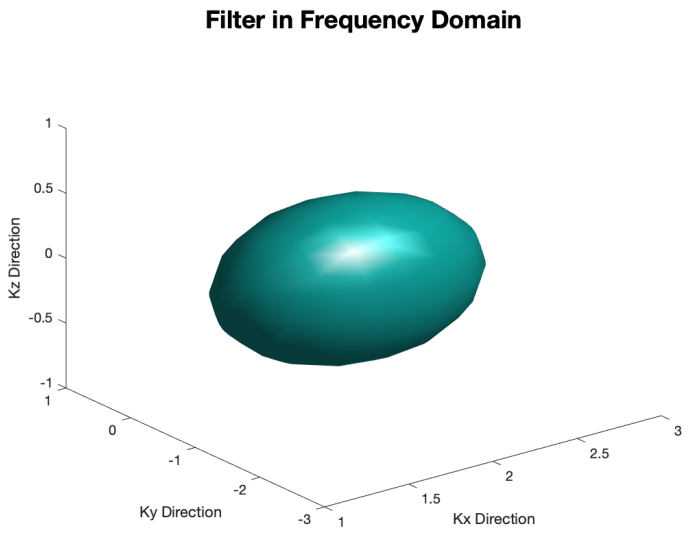


Figure 1. A plot of the 3D gaussian filter. This filter does not have equal sizes in all dimensions, resulting in an oblong shape. A filter reduces values outside of the visualized surface and maintains values within the surface

The second important aspect to understand is filtering. Filtering can work on both frequency domain and time/spatial domain data. Filtering is relatively simple; it is used to reduce the contributions of certain data. In this project, filtering will be used to reduce the noise of the original dataset. Areas of interest will not have their intensity diminished, while everything outside of this range will. To achieve this result, a 3D gaussian filter is used (see Figure 1 for more detail).

III. Algorithm Implementation and Development

As previously explained, this work aims to analyze noisy 3D ultrasound data, and extract an object of interest. The results of this and the code used can be found in the following sections. The primary steps for reaching this goal can be summarized into several steps.

First, the data must be reorganized into an understandable format. Given 64 Fourier modes, we can find that the data can be reconstructed into a 64x64x64 array of values at each time point, for a total of 20 arrays (lines 30 – 33). With this, dimensions for each direction must be formed (lines 14 – 24). In the spatial dimension, values range from -15 to 15 in a 64x64x64 grid in all directions. In the frequency domain, values range from -31 to 32, due to aliasing, in a 64x64x64 grid. However, recall that the frequency is equal to 2π divided by the period. In this instance, the period in all directions is 2L. This means that we must resize all the frequency values by $2\pi/2L$ for proper scaling (line 20). Once finished, the data is now in a comprehensible format.

The next step is to find the object of interest. To do this, several important facts are taken into consideration. Firstly, is that the object will reflect the strongest amplitude frequency (the signature frequency). Secondly, is that when a signal is transformed into the frequency domain, this signature frequency will not change. Although the position of the object may shift in space, the signature frequency will not shift. Thirdly, is that the noise within this data is assumed to be white noise, with zero mean amplitude. This indicates that the noise impacts all frequencies equally, but it averages out to zero over all time points. With these characteristics, locating the object of interest is relatively simple. First, all time points are transformed into the frequency

domain using an FFT, resulting in 20 64x64x64 arrays of frequency. This will align the signature frequency at each time point. Next, these arrays are all averaged together, eliminating much of the noise in the data (lines 30 – 36). Finally, the maximum of the resulting array is found and indexed (line 39). This will return the amplitude and location of the signature frequency. From here, the index found can be traced in the frequency grid created, revealing the values of the center frequency in three dimensions (lines 42 – 47).

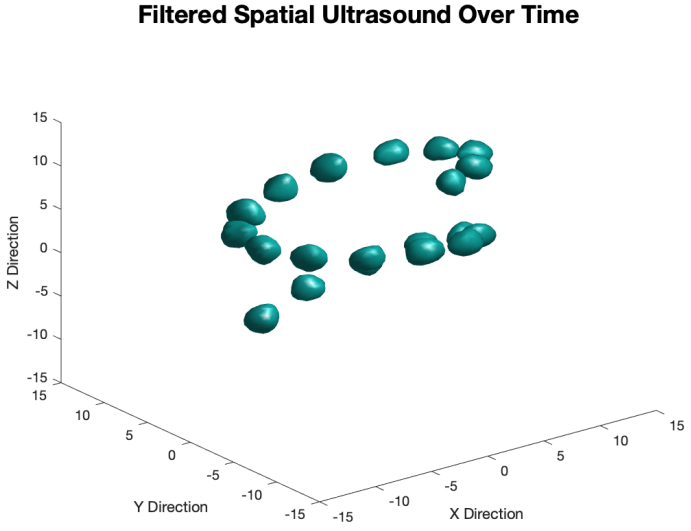
Once the signature frequency has been identified, filtering around this point can be done. This will dramatically reduce the noise, while maintaining the signature frequency. A 3D spectral gaussian filter is first created (line 55, shown in Figure 1). This filter is then multiplied by the frequency domain of the ultrasound at each time point (lines 71 – 77) This data is then transformed back into the spatial domain, using an IFFT (line 80). The result can now be plotted, showing the movement of the object over time (lines 90 – 94).

IV. Computational Results

The attached code in Appendix B show all the details for the steps and plotting of the results shown here. The first step of reorganizing the data allowed for plotting in both space and frequency. However, A plot of this noisy data is not shown, as it is relatively unclear and confusing. Instead, the spectral filter applied to this data (Figure 1), and the visualizations in the space domain are shown (Figures 2 and 3). These plots best illustrate the results of the efforts of this work.

Following the steps above resulted in the successful tracking the object. Figure 2 shows the object as it appears in the ultrasound, moving throughout time (lines 90 – 94). Note that the object at the first time point is at the uppermost (largest Z) spatial point, traveling in a downward spiral. The center of the object is also plotted (lines 99 – 107), which was found by determining the maximum value of the spatial domain at each time point (lines 83 – 87). Figure 3 shows this result, which follows the same trajectory seen previously.

Figure 2. Plot of the object moving through space over time. Note that the first time point occurs at the largest Z value, and the last at the lowest Z value. Appears to follow a downward spiral path.



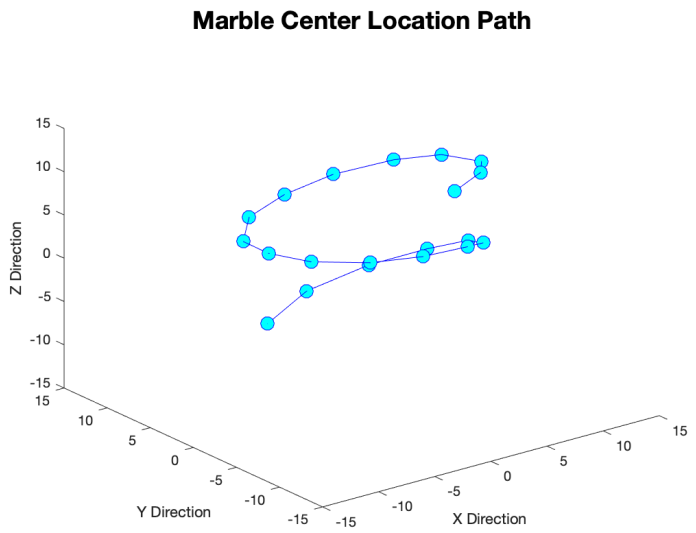


Figure 3. Plot of the path of the object moving through space over time. Each point occurs at the center of the object, as determined by the maximum spatial value. The path in Figure 3 is traced by the lines here. The right-most end point is at the first time point, and the left-most end point is the last time point.

V. Summary and Conclusions

From this work, the practical applications of signal processing and the Fourier transform can be understood. This project involved several important mathematical concepts, such as averaging in the frequency domain, filtering, and understanding the structures of the data. The results highlight how straightforward strategies can solve a daunting problem. Ultimately, the main conclusion of this work is how to apply the Fourier transform and data analysis in meaningful ways.

References

[1] J. Nathan Kutz, *Data-Driven Modeling & Scientific Computation: Methods for Complex Systems & Big Data.* Oxford University Press, 2013.

Appendix A. MATLAB Functions Used

`meshgrid(vec1, vec2, vec3)` – Takes in a set of vectors and returns a coordinate system composed of each. For the purposes of this project, three vectors were used to set the 3D coordinate grid in both space and frequency.

`reshape(vector, size, size, size)` – Takes in a vector and a set of sizes and returns the vector rearranged into those sizes. In this instance, a 1x262144 vector was resized into a 64x64x64 array.

`Fftn(array)` – Computes the FFT of a multidimensional array. In other words, this finds the discrete Fourier transform along each dimension of the given array.

`Max(array,[],'all','linear')` – Finds the maximum value of a given vector. For this situation, this function was used to compute the maximum value across all dimensions and return both the value and corresponding linear index.

`Ind2sub([size size size], index)` – Given an array size and linear index, returns the subscripts of that index. A linear index is the index of an array, if all the data was rearranged

into a single column. Because the max function returns a linear index, this has to be converted into a 64x64x64 subscript so the appropriate location can be found.

Ifftn(array) – Computes the IFFT of a multidimensional array. Functions similar to fftn.

Isosurface(grid1, grid2, grid3, function, contourValue) – Plots the 3D contours of a 4D function, given grids of three dimensions and corresponding values for the fourth. Because the object is given as a function of space, it can't be visualized using a standard plot. Instead, the contour values are plotted, giving a rough estimate of the location of the object.

Plot3(coordinates1, coordinates2, coordinates3) – Plots points or lines, given three dimensional coordinates of each point.

Appendix B. MATLAB Code

```
% Maxwell Weil
% AMATH 482
% 1/24/2020

clear; close all; clc;

% Read in data
load Testdata
L=15; % Spatial domain
n=64; % Fourier modes
trials = 20; % Total number of trials over time

% Setting dimensions for spatial domain
x2=linspace(-L,L,n+1);
x=x2(1:n);
y=x;
z=x;

% Setting frequencies for frequency domain
k=(2*pi/(2*L))*[0:(n/2-1) -n/2:-1];

% Creating full grid of spatial and frequency axes
[X,Y,Z]=meshgrid(x,y,z);
[Kx,Ky,Kz]=meshgrid(k,k,k);

% Initializing average frequency matrix [size of data]
avg_freq_Un = zeros(n,n,n);

% Looping across all time and summing frequencies with reshaped data
for j=1:trials
    Un(:,:,:)= reshape(Undata(j,:),n,n,n);
    avg_freq_Un = avg_freq_Un + fftn(Un);
end

% Calculating average frequency of signal over all time
avg_freq_Un = abs(avg_freq_Un)/trials;

% Finding maximum frequency component (center frequency) and index
[maxval, idx] = max(avg_freq_Un,[],'all','linear');

% Converting linear index to 64x64x64 indices
[row, col, lay] = ind2sub([n n n], idx);
```

```
043
044 % Finding frequency center in 3D frequency domain
045 Kx0 = Kx(row, col, lay);
046 Ky0 = Ky(row, col, lay);
047 Kz0 = Kz(row, col, lay);
048
049 % Setting tau constants for spectral filter
050 taux = 0.2;
051 tauy = 0.1;
052 tauz = 0.3;
053
054 % Creating gaussian spectral filter
055 filter = exp(-taux*(Kx-Kx0).^2-tauy*(Ky-Ky0).^2-tauz*(Kz-Kz0).^2);
056
057 % Plot filter in frequency domain
058 figure(1)
059 isosurface(fftshift(Kx),fftshift(Ky),fftshift(Kz),fftshift(filter), 0.9)
060 title('Filter in Frequency Domain', 'FontSize', 18)
061 axis([1 3 -3 1 -1 1])
062 xlabel('Kx Direction'); ylabel('Ky Direction'); zlabel('Kz Direction')
063
064 % Initializing marble locations matrix [dimensions x time points]
065 marb_loc = zeros(3,trials);
066
067 figure(2)
068 for j = 1:trials
069
070     % Reshaping data at each time point
071     Un(:,:,:)=reshape(Undata(j,:),n,n,n);
072
073     % Change data into frequency domain
074     avg_freq_Un = fftn(Un);
075
076     % Applying spectral filter to frequency data
077     filt_freq_Un = filter .* avg_freq_Un;
078
079     % Transforming data back into spacial domain
080     filt_space_Un = ifftn(filt_freq_Un);
081
082     % Finding maximum spatial point, converting to true axes
083     [maxval2, idx2] = max(filt_space_Un,[],'all','linear');
084     [row, col, lay] = ind2sub([n n n], idx2);
085     marb_loc(1,j) = X(row, col, lay);
086     marb_loc(2,j) = Y(row, col, lay);
087     marb_loc(3,j) = Z(row, col, lay);
088
089     % Plotting spatial contour at each time, animated
090     isosurface(X,Y,Z,abs(filt_space_Un),0.4)
091     title('Filtered Spatial Ultrasound Over Time', 'FontSize', 18)
092     axis([-15 15 -15 15 -15 15])
093     xlabel('X Direction'); ylabel('Y Direction'); zlabel('Z Direction')
094     hold on; drawnow; pause(0.2);
095 end
096
097
098 % Plotting center of marble at each time, animated
099 figure(3)
100 for j = 1:trials
```

```
101     plot3(marb_loc(1,1:j),marb_loc(2,1:j),marb_loc(3,1:j), ...
102         'bo-','MarkerSize',10,'MarkerFaceColor','c')
103     title('Marble Center Location Path', 'FontSize', 18)
104     axis([-15 15 -15 15 -15 15])
105     xlabel('X Direction'); ylabel('Y Direction'); zlabel('Z Direction')
106     hold on; drawnow; pause(0.2);
107 end
108
109 % Display coordinates of final position
110 disp(['Send acoustic pulse to X = ', num2str(marb_loc(1,20)), ...
111     ' Y = ', num2str(marb_loc(2,20)), ' Z = ', num2str(marb_loc(3,20))])
```