Fast Fourier Transform Analysis in Signal Filtering: An Ultrasound Problem

Constance Wang

Abstract

In this project, signal analysis is conducted using the Fast Fourier Transform (FFT). In the problem statement, the goal is to detect a marble traveling through a dog's intestines using ultrasound signals. The marble's environment causes the signal to be very noisy, so FFT must be used in conjunction with filtering techniques to remove noise and locate the marble. The main filtering techniques utilized was FFT averaging and spectral filtering. These techniques, along with some plotting techniques, determined the path of the marble and its final location.

Section I – Introduction and Overview

The Fast Fourier Transform (FFT) is an extremely powerful tool in signal analysis. We use the FFT for signal detection. In this problem, Fluffy the dog swallowed a marble. To locate the marble, the vet suggests using ultrasound to detect spatial variations in Fluffy's intestines. The goal is to trace the movement of the marble in Fluffy's intestines so that an acoustic wave could be focused to break down the marble at its location. This signal, however, is highly noisy because of Fluffy's external movements as well as internal fluid movement in the intestines.

The resulting ultrasound signal is given to us as a 2D matrix. In this matrix, each row is a single point in time. There are 20 rows in the matrix, so the signal was collected at 20 points in time. Each column of the matrix represents a single point in space. There are 262,144 columns, so the signal is taken at 262,144 different points in space. Each point is represented in 3D space, which means that the points in space can be visually represented as a cube of points with 64 points on its edge (64^3 is equal to 262,144). Signal data from each point in space is collected and recorded.

Section II – Theoretical Background

To save Fluffy, we need to understand how FFT was developed and used as a signal analysis tool.

Fourier Transforms, Discrete Fourier Transforms, and the FFT

The motivation behind Fourier Transforms is to take some signal (which can be represented as a function) and break it down into different frequencies (which is represented as sines and cosines of different frequencies). In general, the Fourier Transform can only be used on an infinite domain, as shown by the formula for the Fourier transform below:

$$F(x) = \frac{1}{\sqrt{2\pi}} \int_{-\infty}^{\infty} e^{-ikx} f(x) dx \qquad x \in (-\infty, \infty)$$ Fourier Transform (1)

Complex exponentials are another way to represent sines and cosines, and the function $f(x)$ is the "signal". This equation shows the limitations of Fourier Transforms (as is). This equation works only on the infinite domain, but in cases like Fluffy's , the spatial dimensions exist on a finite domain. The Fourier Transform also can only take functions as an input, however, our signal is collected as discrete points in time.

To overcome these limitations of the Fourier Transform, the Discrete Fourier Transform (DFT) was developed. In this case, the function itself is unknown (as with the case of most signals generated), and the method can be used on finite domains. The DFT formula is given by:

$\hat{x} = \sum_{n=0}^{N-1} x_n e^{\frac{-2\pi i k n}{N}}$ Discrete Fourier Transform (2)

where N is the number of discrete samples taken, and x_n is the discrete values of the signal. The calculation of the DFT is actually an expensive process, which is why it is not often used in performing Fourier Transform analysis. The Fast Fourier Transform used instead because is a much less expensive process, performing the DFT in fewer calculations.

As mentioned in the beginning, FFT can be used as a signal analysis tool. This is because it determines what kind of frequencies are represented in the signal. A plot of FFT plots a discrete set of frequencies against a magnitude measurement. This means that for each frequency, this essentially describes "how much" that frequency is found in the original signal. In the case for Fluffy, the marble itself should have a very distinct frequency character, making it stand out as a peak in an FFT plot. However, there is white noise due to Fluffy's external and internal movements, which can be filtered out using different filtering techniques.

Filter by Average

The reason for using filtering by averaging is because we have multiple samples in time. Since the frequency character of the marble is constant in time, taking the average FFT at different points in time can help filter out the white noise (assuming that the white noise will average out to zero).

Filter using a Spectral (Gaussian) Filter

Filtering using a spectral filter is used when there is a specific, known frequency signal. The filter used in this problem is a Gaussian. This is due to the shape and symmetry of a Gaussian, and the fact that it is centered at a specific point. A 1D example of how spectral filters are used below in Figure 1. The filter function is given by Equation 3. In general, the Gaussian function takes the central (known) frequency as an input. It also takes the width of the Gaussian as a input. The filter works by multiplying the signal by the Gaussian function, which essentially sets everything outside the Gaussian peak to zero, filtering out frequencies that are not of importance to the user.

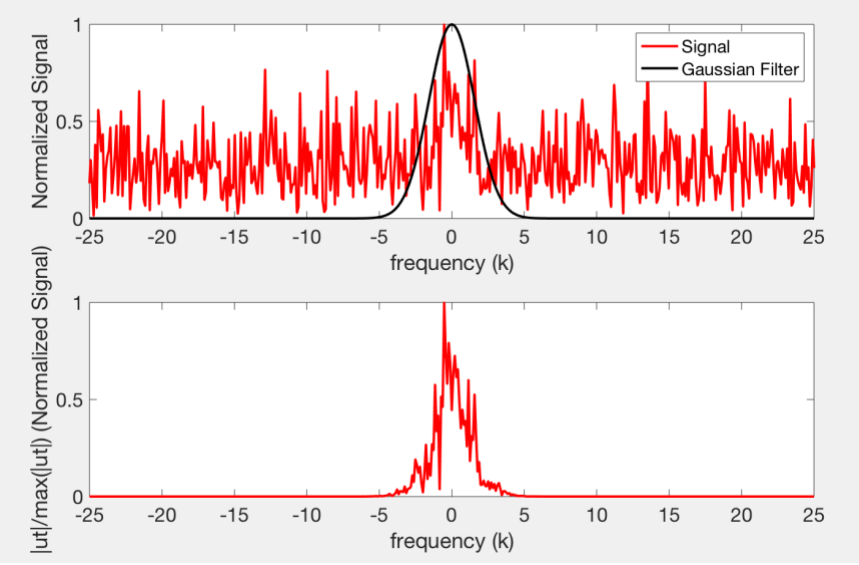

Figure 1: Noisy signal and its associated Gaussian filter function (top). Filtered data after multiplying signal by filter function (bottom).

In Fluffy's case, the marble's frequency character is not known. This means that filtering by spectral filter is not useful, since we don't know what frequency we are looking for. This is why we will first filter by averaging to determine the frequency content of the marble. This specific frequency is then used in the spectral filter, since this filter can better remove the white noise.

Section III – Algorithm Implementation and Development

I split the Matlab code into four different parts, which are four different screenshot references in Appendix B.

Part 1

In this section, the main goal was to initialize any global variables and create the 3D grid coordinates for the spatial and frequency domains. This code was largely unchanged from the reference given from the assignment sample code. The value for n is based on the fact that the signal is 64-by-64-by-64 in 3D (64 points in each dimension), but it is also convenient that this n also satisfies $n = 2^x$ where x = 6. This is because powers of 2 are best for n when using FFT.

In creating the template for the 3D grid coordinates, the domain of discrete values is created using the bounds of L with n points in between. The domain is the same in all dimensions. When using FFT, it will automatically scale all values by 2π, so the domain must be scaled to $\frac{2\pi}{L}$. This is the reasoning behind scaling the variable k (L in this equation is the total length of the domain, which is 2L). In addition, due to the ordering of the FFT (see fftshift( ) in Appendix A), this is the reason for the ordering of k. However, fftshift( ) is used to reorder the domain to be used for plotting purposes.

The first use of the signal data is in the Part 2, however, it is introduced here. The raw data comes as a 20-by-262144 complex matrix (saved as Undata). In the introduction of this paper, this matrix is a linear description of a 3D cube of points. It is better represented as a 64-by-64-by-64 matrix, where the "location" of the point in the matrix is synonymous with a physical location in space. The value of each point in space is the value of the signal data at that point. This "location" can be broken down into x, y, and z coordinates in physical spatial dimensions. To determine the *value* for x, y, and z for this "location" in the matrix, three separate 64-by-64-by-64 matrices (one for each dimension) must be constructed. The "location" in the data matrix is the same location in each of the dimension matrices. This means that at this specific "location" the value of x, y, and z for the signal data matrix is determined. This is the purpose of meshgrid( ): to create these 3D matrices for each dimension. The synonymous approach can be taken with the 3D frequency domain.

Part 2

In this section, filtering by averaging is used. The raw data, however, is in a 2D matrix format, which is not the preferred format for analysis. Instead, a reshape( ) function is used at each point in time. The reshape function takes one row of the raw data and creates a 64-by-64-by-64 matrix out of it, where the "locations" will match that of meshgrid( ). To get the average FFT across the 20 different time signals, a For loop was created to sum all the FFTs across time. To find the true average of the signal in more understandable terms, the absolute value is taken. Notice that in Equation 1 there are complex values represented by the complex exponential in the integral. This is a characteristic of Fourier Transforms in general, but it can make interpreting the results more difficult. The absolute value helps normalize the values to real values.

Notice also that fftshift( ) must be used. This is again due to the property of fft( ), which naturally orders the values starting from the middle. However, in Part 1 we shifted the domain so that it follows the natural low-to-high order, so fftshift( ) must be used to match the order of the domain. This is to accurately determine the location of the frequency signature of the marble. This signature is determined to be the maximum signal in the signal data, since any other signal should be noise. The two functions used for this purpose are described in Appendix A. The key reason for using ind2sub( ) is exactly the reason described in the meshgrid( ) explanation in Part 1: to determine a value at a given "location" within the matrix. This location is described by indices given by ind2sub( ). After knowing this "location" within the matrix, values for the frequency in each dimension can be determined.

Part 3

Determining the value for each frequency determines the central frequency at which the 3D Gaussian filter will be constructed. The reasoning behind this concept is described at the end of Section II – Theoretical Background. This filter is initialized and given by the general equation, Equation 3. k_x, k_y, and k_z are the values of the frequency in each dimension, while k itself is the entire signal data matrix (every value in the matrix is subtracted by respective frequency values). In the loop, the signal must be multiplied by filter for each point in time, which occurs in frequency space. Then, to plot this in the spatial domain, ifftn( ) is used. To plot the location of the marble in space, an isosurface is used. This concept is described in Appendix A. Since it creates surface plots, it gives a good representation of the marble in space.

The path of the marble can be determined again using the max command. This is because the marble has the most frequency character in both the spatial and frequency domains. Again, the "location" can be determined, and the x, y, and z values of the marble at each point in time are saved to generate a plot3 (see Appendix A) to show the path of the marble. The points are saved, so the final location of the marble can be recorded.

Part 4

Coordinates are saved for plot3, so the last x, y, and z coordinates are the coordinates of the final marble position. These are with respect to the location of the middle of the signal.

Section IV – Experimental Results

Frequency Signature using Averaging

The FFT of the signal data was taken for each of the 20 points in time. The resulting FFTs for each point in time were summed together and averaged. It is assumed that the maximum signal from the averaged FFTs is the frequency signature of the marble. This frequency was determined in the x, y, and z dimensions, which is used in the second part of the solution. These frequencies are as follows:

$$x\ frequency = 1.8850$$
$$y\ frequency = -1.0472$$
$$z\ frequency = 0$$

Marble Location and Path using Gaussian Filter

Using the frequencies determined from the previous part, a Gaussian filter in 3D was constructed:

$$filter = e^{-\tau((k-k_x)^2(k-k_y)^2(k-k_z)^2)}$$ 3D Gaussian Filter Function (3)

Where k_x, k_y, and k_z are the frequencies in the x, y, and z directions respectively, and τ is 0.2 (constant determined empirically). This FFT of the original signal data was multiplied by the Gaussian filter for each point in time. An isosurface plot was generated after taking the inverse transform of the filtered data, showing the marble location in spatial dimensions. The location of the marble determined by finding the maximum signal (of the filtered data) and its location in x, y, and z space at each point in time. Using plot3, a path can be drawn from the first point in time to the last point in time the marble was located. Both the isosurface plots (for each point in time) and the marble path plots are shown in Figure 2 below.

The final location of the marble was saved, which is determined to be the location at which an intense acoustic wave could be focused to break down the marble. This location with respect to the middle of the ultrasound signal is:

$$x\ final\ location = -5.6250$$
$$y\ final\ location = \ 4.2188$$
$$z\ final\ location = -6.0938$$

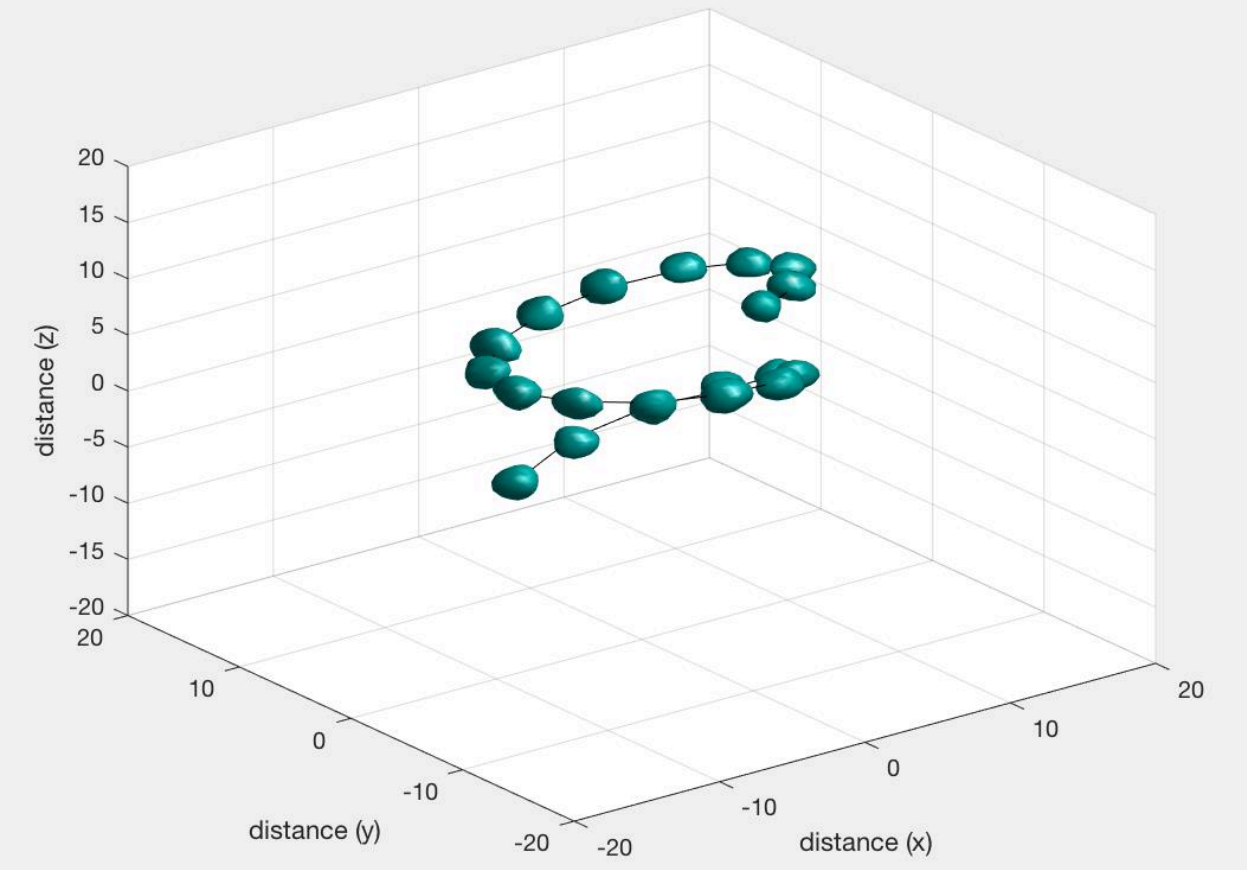

Figure 2: Final plot of the marble location (surface plots) and the marble path (line that passes through each surface). There are 20 surface plots corresponding to the 20 points in time.

Section V – Summary and Conclusions

The location of the marble can be determined using FFT in multidimension space. This is because the FFT is a tool that can determine the frequency character of the signal, or, how much of each frequency exists in the original signal. In this problem, the marble that was stuck in Fluffy's intestine has a unique frequency character since the rest of the surrounding environment gave off only noisy signals. It was this frequency character that allowed FFT and filter techniques to be powerful in determining the location of the marble. In this analysis, the FFT proved to be a powerful tool because the ultrasound was stationary. However, FFT breaks down if the ultrasound was moved in time, since location can only be quantified with respect to the center of the ultrasound. However, this analysis shows that FFT is not only a quick processing tool, but also a tool that can be used for signal analysis.

Appendix A

- **fftn( )** – takes the Fast Fourier Transform in n-dimensional space
    - Used in the first and second parts of the problem to determine the frequency character of the ultrasound signal, since the frequency character of the marble differs from that of the noise
- **ifftn( )** – takes the Inverse Fourier Transform in n-dimensional space
    - Used to convert the frequency data back into spatial data so that the precise location of the marble in space could be determined
- **fftshift( )** – shifts the values of the FFT to match those regularly used (usually for plotting purposes)
    - One property of the FFT command in Matlab is that it will shift the data to $x \in [0, L] \rightarrow [-L, 0]$ and $x \in [-L, 0] \rightarrow [0, L]$
    - This means that the FFT must be shifted to match the sequence of values of the frequency domain
- **meshgrid( )** – used to create a 3D grid coordinate arrays for our x, y, z spatial axes and the respective axes in the frequency domain
    - This is primarily used to match the signal or frequency coordinates of the reshaped signal data
- **reshape( )** – used to create a 3D array (it uses one row of the 20-by-262,144 matrix of signal data points) to represent the 3D nature of the signal data for *each* time point
- **[M, I] = max(U(:))** – used to find the maximum value, and the index in the matrix this maximum values occurs
    - Used to determine the frequency signature of the marble, as well as its location in spatial dimensions
- **ind2sub( )** – the index of given by the max command is in linear coordinates, so this command changes the index to a 3D index location so that we can locate this value within each of the spatial and/or frequency dimensions
- **isosurface(X, Y, Z, V, isovalue)** – given values of V, this creates the plot of the marble in the space determined by the X, Y, and Z coordinates at some specific isovalue, which allows visualization of 4D data in 3D (you "set" one of the dimensions to the constant, isovalue)
    - isovalue is an empirically determined value
    - a surface plot is created between every isovalue value
- **plot3( )** – plots a set of coordinates in 3D connected by lines between each coordinate
    - Used to plot the path of the marble