3D Spatial Fourier Analysis

Joe Henthorn
University of Washington

January 2020

Abstract: The goal of this paper is to demonstrate a useful application of the Fast Fourier Transform (FFT) to analyze 3D spatial data. This paper also explains how to filter a noisy multidimensional signal with a Gaussian filter and to visualize the results.

1 Introduction

The follow document is a guide for using the multidimensional Fast Fourier Transform and constructing a frequency filter. The accompanying MATLAB code is a guide for using the FFT to find an objects trajectory in 3D space. The data used in this example is hypothetical ultrasound spatial data from a dog that has eaten a marble. The trajectory of the marble is tracked as it passes through the canine alimentary canal. The MATLAB code also demonstrates how to make pretty plots and a gif animation of the object's trajectory.

2 Theoretical Background

Review: The Fourier Transform is a mathematical tool that enables the frequency analysis of a particular function or set of data. The Transformation maps time or spatial domain data to the frequency domain with the integral of the product of a function with a complex exponential function. The critical connection that makes this transformation possible is Euler's number $\boldsymbol{e}$ and Euler's formula:

$$e^{i\theta} = cos\theta + isin\theta \tag{EQ:1}$$

From this identity, we can describe any function or data signal as a sum of cosine and sine functions, such that their period corresponds to the exponential argument of the transform. The following formulas are the continuous Fourier Transform and the inverse Fourier Transform.

$$Fourier \quad Transform$$
$$\hat{f}(\omega) = \int_{-\infty}^{\infty} f(x) \cdot e^{-2\pi i\omega} dx \tag{EQ:2}$$

$$Inverse \quad Fourier \quad Transform$$
$$f(x) = \int_{-\infty}^{\infty} \hat{f}(\omega) \cdot e^{2\pi i\omega} d\omega \tag{EQ:3}$$

For discrete values we use the discrete Fourier transform that use a sums of discrete points over a finite interval instead of an infinite integral. These are the equations we use for our numerical methods.

$$Discrete \quad Fourier \quad Transform$$
$$X_\omega = \sum_{n=1}^{N-1} x_n \cdot e^{-2\pi i\omega/N} d\omega \tag{EQ:4}$$

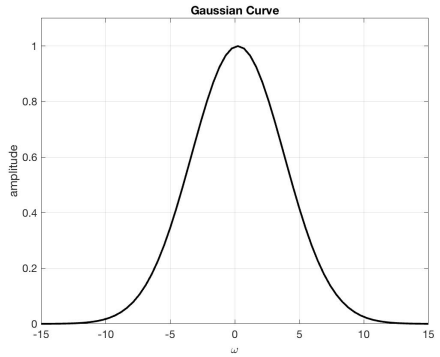
$$Discrete \quad Inverse \quad Fourier \quad Transform$$
$$x_n = \frac{1}{N} \sum_{\omega=0}^{N-1} X_\omega \cdot e^{-2\pi i\omega n/N} d\omega \tag{EQ:5}$$

3 Methods & Algorithm Implementation

For this example, we use the built in MATLAB function **fftn()** and **ifftn()** to transform the n dimensional ultrasound data. In this particular example, the spatial data is very noisy and needs to be cleaned up for analysis. First the spatial domain data is unpacked and structured into an n x n x n matrix with **reshape()**, where n is the number of Fourier nodes. A frequency vector is created for our dependant variable axis in the frequency domain, and it is scaled by the factor π/L, where L is the number of data points. Then the signal data is transformed with the **fftn()** function. This is done over a for loop for each sample of data. We also use the **meshgrid()** function for building a mesh of points for our spatial and frequency data. Finally, we utilize the **fftshift()** command to center our frequencies spectrum.

All of the frequency domain data is averaged and then inspected to find the frequencies with the highest amplitudes for each frequency spectra. This is achieved with the **[Value Index]=max()** function. We can then use these max values to find their corresponding frequency value with the **ind2sub(Index)** function. The frequencies corresponding to the highest amplitude are then used to construct our Gaussian filter and suffice as our k0 values. The lower value areas of the Gaussian curve attenuate all signal in those areas, while the higher values of the Gaussian curve preserves the signal (fig. 1). The Gaussian filter is constructed by the following mathematical expression.

$$Gaussian \quad Filter \quad Function$$
$$\mathcal{F}(\omega) = e^{-\tau(\omega-\omega_0)^2} \tag{EQ:6}$$

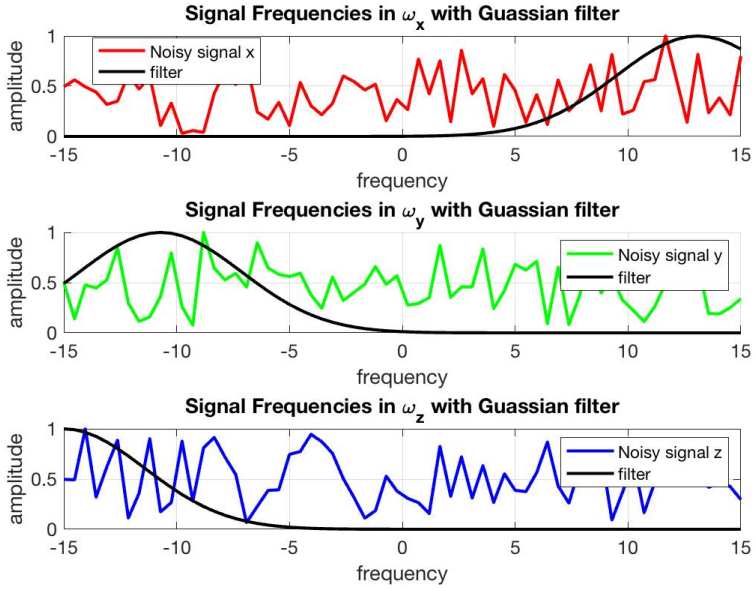


.

Figure 1: Plot of Gaussian function curve centered at zero, ω is frequency.

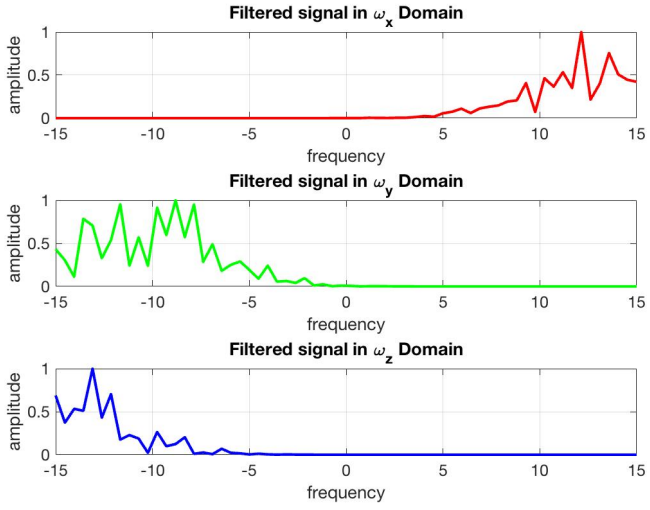
Additionally, the value we pick for τ determines the width of our filter. Lower values of τ produce wider filter windows, while higher values produce a narrow filter window.

Next, the filter is applied to the noisy signal (fig. 2). The result is a filtered signal in the frequency domain with only the frequencies of interest (fig. 3). These particular frequencies of interest represent the marble passing through the dog's intestines.



.

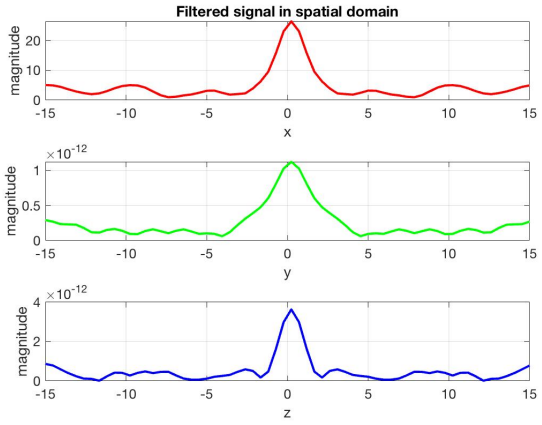
Figure 2: Plot of noisy frequency domain signal for each dimension with the corresponding Gaussian filter overlay at the maximum amplitude frequency.



.

Figure 3: Plot of filtered signal(s) in the frequency domain.

Finally, the clean frequency signal data is transformed back into the spatial domain with the inverse Fourier transform. To achieve this, we use the **ifftn()** function. The spatial data is now cleaned and easier to analyze (fig. 4).

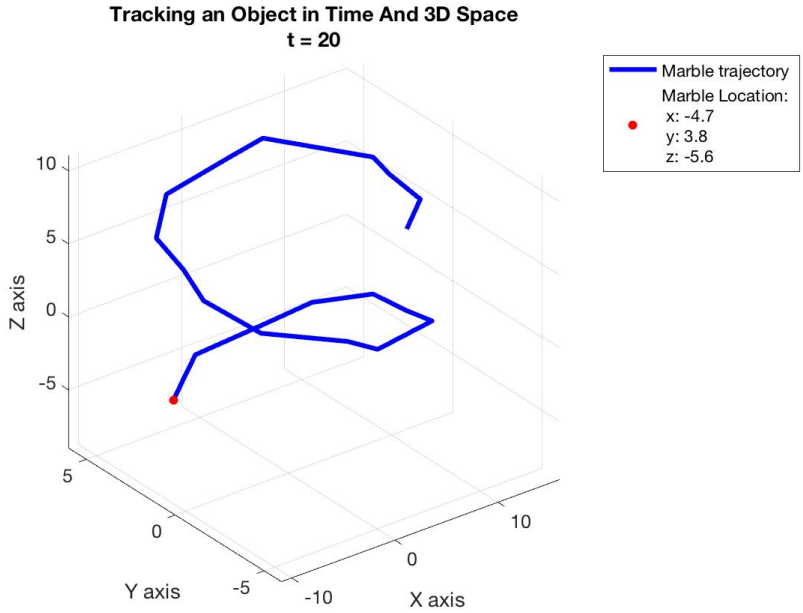


.

Figure 4: Plot of filtered signal(s) in the spacial domain.

4 Computational Results

In this example it is easy to see the trajectory and location of the object (marble) for each sample. The final location of the marble at the last sample point was found to be (-4.6875 3.7500 -5.6250) (fig. 5).



.

Figure 5: Plot of marble trajectory (in blue) and final location (in red) at sample 20.

5 Conclusion

In this example we used noisy spatial data and cleaned it up in frequency space. We used the Fourier transform to get the frequency representation of our data, and we used the inverse Fourier Transform to relate our filtered data back into the spatial domain.