AMATH 582 Homework 1

Luke Zhu, 1923117

January 24, 2020

Abstract

Fourier transform converts time series or spatial data into the frequency domain. This lends itself extremely useful in the field of signaling processing. In particular, the use of Fourier transformation and a frequency filter can effectively differentiate signal from noise. Additionally, averaging multiple Fourier transformed measurement data can effectively denoise the data when the noise is normally distributed. Here in this report, the author was presented with a case where a dog swallowed a marble and acoustic wave is needed to help break up the marble. Fourier transform and the aforementioned techniques to denoise was applied to the measurement data gathered through ultrasound. Ultimately, the author was able to determine the marble's path in the dog's intestine and offered suggestion on where the acoustic wave should target in order to destroy the marble based on its location at the last imaging time point.

1 Introduction and Overview

Fourier transformation is commonly used in the field of signaling transformation because of its ability to convert a time or spatial domain signal into its frequency counterpart. This property, coupled with several other denoising techniques, such as applying a frequency filter or averaging multiple fourier transformed data measurements, allows for reliable differentiation of signal and noise presented in the data. In this assignment, the author was tasked with determining the path of a marble that was trapped in the dogs intestine. The ultrasound equipment used for imaging has a spatial domain of $[-15, 15]$ in the x, y and z direction, and is able to resolve to 64 data points along each direction. Due to the noise inherent in the data measurement, Fourier transform was required to clean up the data. The key theoretical background of Fourier transform and its associated data processing techniques were expanded upon in detail in the following section. The ensuing section would give a high level overview of how the algorithm was implemented to solve this specific problem. Lastly, the computation results were presented and the precise location of the marble was determined at the last measurement point, which would enable the use of an intense acoustic wave to break up the marble.

2 Theoretical Background

As we learned from our textbook, Fourier introduced the concept of representing a given function $f(x)$ by a trigonometric series of sines and cosines:

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

Extending this idea to the continuous domain, the Fourier Transform and its inverse is defined as

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

$$f(x) = \frac{1}{\sqrt{2\pi}} \int_{-\infty}^{\infty} e^{ikx} F(k) dk \tag{3}$$

where k is the wave number, and x could be either representing a spatial or temporal variable. Note that the transformation is over the entire real line. In most practical applications, the transformation is only applied to a finite domain $[-L, L]$ and the data is often discretized. Thus the Fast Fourier Transform (FFT) algorithm was developed in the 1960s to speed up the computation. This algorithm was able to achieve a blazingly fast asymptotic runtime of $O(NlogN)$, where N is the data size. However, there are two caveats associated with this algorithm. First the algorithm assumes a 2π periodic signal, thus the wavenumber in the algorithm must be rescaled by the factor $2\pi/L$ to accommodate for different domains. Secondly, the FFT algorithm would shift the discretization domain, such that $x \in [0, L] \rightarrow [-L, 0]$ and $x \in [-L, 0] \rightarrow [0, L]$. Similarly, when performing an inverse FFT, the frequency signal must also be shifted before undergoing the transformation. Therefore the FFT and IFFT function is generally accompanied by another function that undoes the shift for the ease of computation and human interpretation. In the MATLAB toolbox, the FFT can be performed with `fft(x)` and in the case of higher dimension transformation, `fftn(x)`. Conversely, the reverse FFT can be performed with `ifft(x)` and `ifftnx`. The shifting operation can be performed with `fftshift(x)`.

One of the bigger challenges in the signal processing field is how to reliably differentiate noise from signal. Fourier transform is especially well suited for this task as it allows for the transformation of a signal into the frequency domain, which further facilitates the extraction of information contained within a specific frequency range. This is commonly done by building a filter that is centered around the frequency of interest. In practice, the continuous and smooth properties of the Gaussian distribution lends itself useful in fitler applications. More formally, a Guassian filter is defined as

$$\mathcal{F}(k) = exp(-\tau(k - k_0)^2) \tag{4}$$

where τ is the bandwidth of the filter, k is the wavenumber and k_0 is the center frequency of the desired signal field. The bandwidth controls the range of signals to filter, a higher bandwidth value corresponds to a larger range of frequencies. A common initial value is 0.2.

Moreoever, if we assume that the noise presented in the data is normally distributed with zero mean and unit variance, then by averaging across many measurements and signals, the average noise should converge to 0. Doing so essentially denoises the data and improves the ability to discriminate between noise and signal. However, complication arises when the signal is varying in the either the time or spatial domain. The signal no longer occurs at the same time stamp or spatial location, thus the simple averaging technique would no longer work. But if its underlying frequency signature remains constant, then it would become beneficial to first transform the measurement data into the frequency domain using Fourier transformation and then averaging out the transformed data.

In combination, building a filter and averaging the signal across multiple measurements allows for reliable discrimination between noise and signal. Both these techniques would be employed in this assignment to demonstrate their utility in signaling processing.

3 Algorithm Implementation and Development

The algorithm on how to determine the frequency signature generated by the marble and denoise the data based on that signature is expanded upon in this following section.

Algorithm 1: Algorithm to determine the frequency signature generated by the marble

```
Import data from Testdata.mat
Intialize 3D matrix $Utnp\_avg$ to store the averaged result of the signals
for $j = 1 : 20$ do
    Extract measurement $j$ from the raw data, reshape it to a $n \times n \times n$ matrix
    Perform multi-dimension FFT and fftshift on the reshaped measurement data
    Add the result to $Utnp\_avg$
end for
$Utnp\_avg \leftarrow Utnp\_avg/20$
Normalize $Utnp\_avg$
for $i$ = 0:1:0.1 do
    Plot $Utnp\_avg$ with isosurface at different isovalues
end for
Determine the isovalue that revealed the frequency signature
```

Algorithm 2: Algorithm to denoise the data and determine the path of the marble

```
Initialize an array to store the 3D path of the marble
for $i = 1 : 20$ do
    Initialize the center frequency $f_x$, $f_y$ and $f_z$ to values determined in the previous algorithm
    Set the bandwidth parameter $\tau$
    Create a three dimensional unshifted Gaussian filter based on the bandwidth parameter and center frequency
    Extract measurement $i$ from the raw data, reshape it to a $n \times n \times n$ matrix
    Perform FFT on the measurement and then fftshift to unshift the result
    Apply the Gaussian filter to the result
    Apply fftshift to the filtered signal in preparation for IFFT
    Apply inverse FFT to convert the signal back to spatial domain
    Extract the $x$, $y$, $z$ location that corresponds to the highest signal and store it in the 3D path array
end for
Plot the 3D path of the marble
```

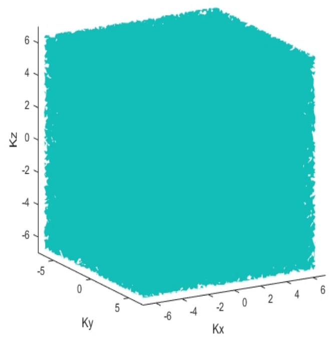
4 Computational Results

A 3-dimensional Fourier transform was first applied to the data and the transformed signals were averaged and then normalized. To determine the frequency signature, the result is plotted and visualized through the MATLAB isosurface function at various isolevels (see Figure 1). At an isolevel of 0.7, the center frequency signature became apparent and the values are tabulated in Table 1.

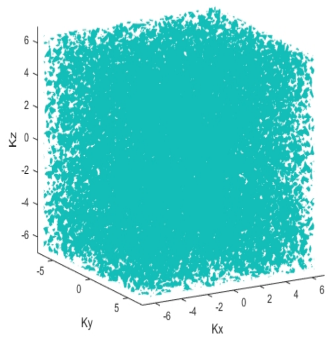
Table 1: Tabulated results of the center frequency

Central Frequency	f_x	f_y	f_z
Value	2	-1	0

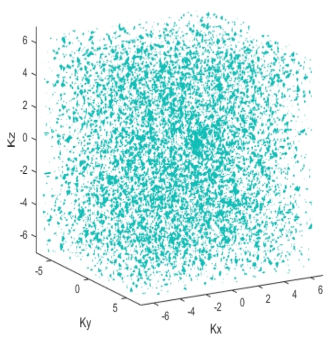
Subsequently, a denoising step was applied to each and every time slice with a Gaussian filter centered at the frequency values tabulated in Table 1 and a bandwidth parameter (τ) of 0.2. The filtered signal was then transformed back into the spatial domain. The spatial location that corresponded to the highest signal value was used to approximate the location of the marble at each time step. The trajectory of the marble is shown in Figure 2. It was observed that the marble followed a downward spiral pattern down the dog's intestine. This also suggested that the intense acoustic wave should be focused on the location $x = -5.63, y = 4.22, z = -6.09$ at the 20th data point.



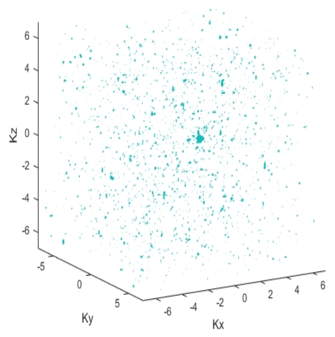
(a) normalized isovalue = 0.2



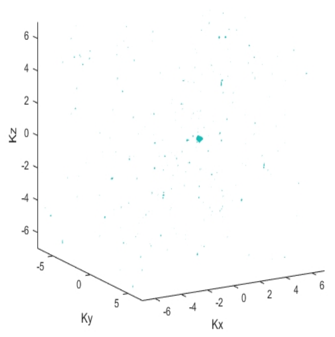
(b) normalized isovalue = 0.3



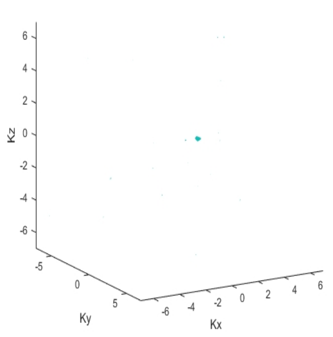
(c) normalized isovalue = 0.4



(d) normalized isovalue = 0.5



(e) normalized isovalue = 0.6



(f) normalized isovalue = 0.7

Figure 1: Normalized isosurface plot of the fourier transformed signal after averaging all measurements

5 Summary and Conclusions

In summary, by fourier transforming the imaging data into the frequency domain and averaging out the measurements, the noise was effectively filtered out, and the frequency signature was identified at $f_x = 2, f_y = -1, f_z = 0$. A Gaussian filter was then applied to the each measurement data centered around the center frequency signature. As such, the position of the marble was detected at each time slot. Plotting the path of marble revealed that it followed a downward spiral pattern. Lastly, it was determined that in order to break up the marble at the 20th data measurement, the intense acoustic wave should be aimed at the location $x = -5.63, y = 4.22, z = -6.09$.

Appendix A MATLAB Functions

This section includes some of the important MATLAB functions that were used to process the data along with a brief implementation explanation.

- `y = linspace(x1,x2,n)` returns a row vector of `n` evenly spaced points between `x1` and `x2`.
- `[X,Y] = meshgrid(x,y)` returns 2-D grid coordinates based on the coordinates contained in the vectors `x` and `y`. X is a matrix where each row is a copy of `x`, and `Y` is a matrix where each column is a copy of `y`. The grid represented by the coordinates `X` and `Y` has `length(y)` rows and `length(x)` columns.
- `Y = fftn(X)` returns the multidimensional Fourier transform of an N-D array using a fast Fourier transform algorithm. The N-D transform is equivalent to computing the 1-D transform along each dimension of X. The output Y is the same size as X.

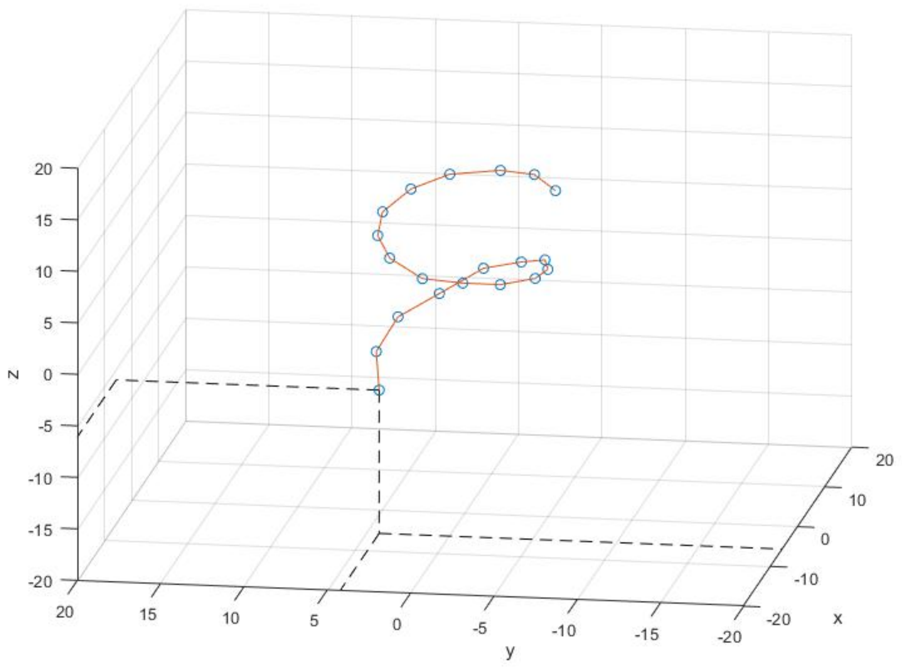


Figure 2: The marble trajectory in the dog's intestine

- `X = ifftn(Y)` returns the multidimensional discrete inverse Fourier transform of an N-D array using a fast Fourier transform algorithm. The N-D inverse transform is equivalent to computing the 1-D inverse transform along each dimension of Y. The output X is the same size as Y.
- `Y = fftshift(X)` rearranges a Fourier transform X by shifting the zero-frequency component to the center of the array.
- `B = reshape(A,sz)` reshapes A using the size vector, sz, to define size(B). For example, reshape(A,[2,3]) reshapes A into a 2-by-3 matrix. sz must contain at least 2 elements, and prod(sz) must be the same as numel(A).
- `fv = isosurface(X,Y,Z,V,isovalue)` computes isosurface data from the volume data V at the isosurface value specified in isovalue. That is, the isosurface connects points that have the specified value much the way contour lines connect points of equal elevation.
  The arrays X, Y, and Z represent a Cartesian, axis-aligned grid. V contains the corresponding values at these grid points. The coordinate arrays (X, Y, and Z) must be monotonic and conform to the format produced by meshgrid. V must be a 3D volume array of the same size as X, Y, and Z.
  The struct fv contains the faces and vertices of the isosurface, which you can pass directly to the patch command.

Appendix B MATLAB Code

GitHub account username: **Lukez-pi**
GitHub assignment repository: **AMATH582**

```
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);
Utnp_avg = zeros(n, n, n);

% converting each signal into its frequency domain and averaging the
% signals
for j=1:20
Un(:,:,:)=reshape(Undata(j,:),n,n,n);
Utn = fftn(Un);
Utnp = fftshift(Utn);
Utnp_avg = Utnp_avg + Utnp;
end
Utnp_avg = Utnp_avg / 20;

% plot the averaged fourier transformed signal at different isolevels
fig_counter = 1;
for level = 0.1:0.1:0.9
    figure(fig_counter)
    axis([-7 7 -7 7 -7 7])
    isosurface(Kx, Ky, Kz, abs(Utnp_avg)/abs(max(Utnp_avg(:))), level);
    view(30,-15)
    xlabel('Kx'), ylabel('Ky'), zlabel('Kz')
    set(gcf,'color','w');
    imagewd = getframe(gcf);
    imwrite(imagewd.cdata, num2str(fig_counter)+".jpeg", "Quality", 100);
    fig_counter = fig_counter + 1;
end

% the center frequency
f_x = 2;
f_y = -1;
f_z = 0;

tau = 0.2;
filter = exp(-tau*((Kx - f_x).^2 + (Ky - f_y).^2 + (Kz - f_z).^2));

x_coord = zeros(1, 20);
y_coord = zeros(1, 20);
z_coord = zeros(1, 20);

%filter each measurement in the frequency domain and perform inverser
%fourier transform
for j=1:20
```

```
Un(:,:,:)=reshape(Undata(j,:),n,n,n);
Utn = fftn(Un);
Utnp = fftshift(Utn);
Utnp_f = filter .* Utnp;
Unp_f = ifftn(fftshift(Utnp_f));
[~, idx] = max(Unp_f(:));
x_coord(j) = X(idx);
y_coord(j) = Y(idx);
z_coord(j) = Z(idx);
end

% plotting the 3D trajectory of the marble
figure(1)
plot3(x_coord, y_coord, z_coord, 'o')
hold on
plot3(x_coord, y_coord, z_coord)
hold on
plot3([x_coord(end), x_coord(end)], [y_coord(end), y_coord(end)], [z_coord(end), -20], '--
plot3([x_coord(end), x_coord(end)], [y_coord(end), 20], [z_coord(end), z_coord(end)], '--k
plot3([x_coord(end), -20], [20, 20], [z_coord(end), z_coord(end)], '--k')
plot3([x_coord(end), x_coord(end)], [y_coord(end), -20], [-20, -20], '--k')
plot3([x_coord(end), -20], [y_coord(end), y_coord(end)], [-20, -20], '--k')
axis([-20 20 -20 20 -20 20]), grid on, drawnow
xlabel('x'), ylabel('y'), zlabel('z')

% the spatial location of the marble at the last measurement
xf_coord = x_coord(20)
yf_coord = y_coord(20)
zf_coord = z_coord(20)
```