AMATH 482 HW 1

Anh-Minh Nguyen

January 13, 2020

Abstract In this report, we will use a version of the Fourier Transform, along with Gaussian Filtering and Averaging to locate the trajectory of a marble through a dog's intestines. We use MATLAB and its functions to carry out the algorithms.

1 Introduction and Overview

My dog, Fluffy, has unfortunately swallowed a marble, and it has worked it's way into my dog's intestines. Data obtained using ultrasound points to spacial variations in a small area of the intestines. Because my dog keeps moving, the internal fluid movement generates highly noisy data when the veterinarian uses ultrasound to attempt to locate the exact position of the marble. To save my dog's life, we will locate and compute the trajectory of the marble using Fast Fourier Transform to process the data. In addition, we will use methods such as Filtering and Averaging to mitigate the noisiness of our data.

2 Theoretical Background

To begin, Fourier Series is the process taking a signal, in this case the ultrasound data, and essentially decomposing it into sines and cosines of different frequencies. Thus, we are taking of function of space and time and translating it into a function of frequency. It is important to note that $f(x)$ is an integrable, periodic function. Within the interval, $[-\pi, \pi]$, we have that $\int_{-\pi}^{\pi} |f(x)|dx < \infty$. If $f(x)$ is discontinuous, it must also be single-valued, monotonic piecewise continuous, containing a finite number of extrema and jumps. We usually denote the frequency as k and the space or time as t or x.

$$f(x) = \frac{a_0}{2} + \sum_{k=1}^{\infty}[a_k cos(kx) + b_k sin(kx)]$$

The coefficient terms a_k and b_k are called Fourier coefficients, and they determine the weight of each sine and cosine term.
The Fourier Transform can be seen as an extension of the Fourier Series as it can be applied to non-periodic functions of space and time. The interval is broadened to $[-\infty, \infty]$ for both x and k.

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

The inverse Fourier Transform reverses the process.

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

However, we have 20 measurements, a discrete amount of values. Thus, we introduce the Discrete Fourier Transform (DFT). This transform converts equally-spaced values of time or space into equally-spaced values in the Fourier domain. The DFT has computing complexity of $O(N^2)$, where N is the size of our data. So, we have a faster version called the FFT, which has complexity $O(NlogN)$.
One of the main problems we come across though is noise. Noise is variation that can masks the true nature of signals. Two methods we can use to help us include Filtering and Averaging.

First, Filtering is the process of extracting information through denoising, but it requires we know the frequency of interest. This allows us to specify the width of our filter and frequency in general. We will use the Gaussian filter:

$$F(k) = e^{-\tau(k-k_0)^2}$$

We multiply our frequencies with this function, and as we see, anything within the filter will be left close to it's original frequency while anything farther from our frequency of interest will go to zero.
Second, Averaging is useful in that it helps address some weaknesses of Filtering, especially that the noise is assumed to be white and that signals can be continually produced. We can model white noise by adding a normally distributed random variable with zero mean and unit variance to each Fourier component of the spectrum. So, if we add multiple signals together, the expected value of the combined white noise should go to 0. Then, we normalize the combined signals. This must be done in the Fourier domain because it produces a localized signature since the spectrum remains fixed at the transmitted signal frequency, and if it is done in the time domain, we will be confounded by the time component. These mathematical techniques can help us compute the trajectory of the marble within our dog's intestines.

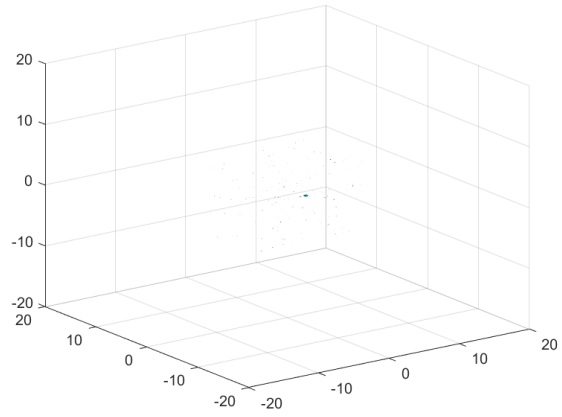
3 Algorithm Implementation and Development

To begin, we specify the spacial domain, $L = 15$. Then since we are using the discrete version of the Fourier Transform, we choose a number, that is equivalent to 2 raised to a positive integer, of evenly spaced points. In this case, we choose $N = 64$ points, starting from -15 and ending just before 15 because of periodicity. Since our data is 3-D, we designated the same point schemes for our x, y, and z dimensions.
We have to re-scale our frequencies by multiplying them by $\frac{2\pi}{L}$ because the FFT assumes 2π periodic frequencies due to the nature of the sine and cosine functions and our space domain length. Next, we want to center the highest frequency in the middle, so we place $-N/2$ in the middle and arrange the point scheme to fit it. We will use *fftshift* function to shift our transformed function back to its mathematically correct positions, and we preemptively do that to our Fourier modes, k, which we then label it as ks. Last, we use MATLAB's meshgrid function to create the grids for plotting purposes.
We will start with Averaging the signals in the frequency domain. By using a for loop, we can reshape each measurement into a 64x64x64 matrix and use our multi-dimensional Fourier Transform on it. Then, we can add each Fourier-transformed signal together and divide it by m, our number of signals. Afterwards, we grab the absolute value, divide our averaged signal by it's maximum value, and shift it to display it properly on a plot. We can then find the index of our largest frequency and use it for filtering.
Note, in our case, we have to create a 3-D filter. In essence, we will be applying a filter for each dimension, so we are multiplying each signal by every filter, which we can write as a single exponential. Once again, we will shift our filter with *fftshift* because when calculating the index, recall we were using the non-shifted (or shifted twice) frequency values, ks.Finally, for each signal, we can apply the filter and perform the inverse Fourier Transform to locate the trajectory of the marble. To plot in 3D, we grab the largest signal of each measurement as it's likely to be our marble.

4 Computational Results

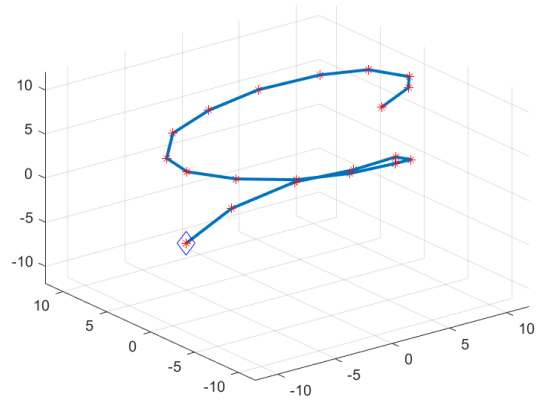
After averaging, these were our results.

Figure 1: Averaged Isosurface, isoValue = 0.6



The matrix coordinates of the frequency of interest within the averaged signal is $(28, 42, 33)$.
Then, we can figure out our frequency of interest within the Fourier domain, and we end up with roughly $(1, 8850, -1.0472, 0)$.
Using this, we can apply our filter and determine the trajectory.

Figure 2: Trajectory of Marble



Finally, we end up with the final position, in the time domain, being $(-5.6250, 4.2188, 6.0938)$.

5 Summary and Conclusions

We see that we were able to predict the trajectory and current location of the marble using Fourier Transforms and supplemental denoising techniques. With the marble hovering around (-5.6, 4,2, 6.1), the vet will know be able to develop a plan to remove the marble from our dog, Fluffy.

6 Appendix

A MATLAB Functions

- *fftshift*: Switches opposite sides of a data object

- *fftn*: Fourier Transform of an n-dimensional object
- *squeeze*: Reduces dimension of an n-dimensional object by removing length 1 dimensions

B MATLAB Code

```
clear; close all; clc;
load Testdata
L=15; % spatial domain
n=64; % Fourier modes
leng=262144;
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);
%% Average together 20 noisy signals in the Fourier domain
m = 20;
ave = zeros(n, n, n);
UndataFT = zeros(m,n,n,n);

for j=1:m
    UndataFT(j,:,:,:) = fftn(reshape(Undata(j,:),n,n,n));
    ave = ave + squeeze(UndataFT(j,:,:,:));
end
aveft = abs(fftshift(ave))/m;
aveft = abs(aveft./max(abs(aveft), [], 'all'));

%% Graph the averaged frequencies
close all; clc;
figure(1);
isosurface(Kx,Ky,Kz, aveft,0.6)
axis([-20 20 -20 20 -20 20]), grid on
%% Find the index of max value from averaging & apply filter
[mm, index] = max(aveft(:));
[px,py,pz] = ind2sub([n,n,n], index); % 28, 42, 33

tau = 0.2;
k0x = Kx(px,py,pz); % 1.8850
k0y = Ky(px,py,pz); % -1.0472
k0z = Kz(px,py,pz); % 0
filter = exp((-tau*(Kx-k0x).^2) + (-tau*(Ky-k0y).^2) + (-tau*(Kz-k0z).^2));
filter = fftshift(filter); % Recall that our values need to be shifted
marbleTrajectory = zeros(m,3);
for j=1:m
    UndataS = ifftn(filter.*squeeze(UndataFT(j,:,:,:)));
    [mmm, indx] = max(UndataS(:));
    [mx, my, mz] = ind2sub([n,n,n], indx);
    marbleTrajectory(j,1) = X(mx,my,mz);
    marbleTrajectory(j,2) = Y(mx,my,mz);
    marbleTrajectory(j,3) = Z(mx,my,mz);
end

%% Graph trajectory of marble
clf; close all;
```

```
figure(2)
plot3(marbleTrajectory(:,1), marbleTrajectory(:,2), marbleTrajectory(:,3), 'LineWidth', 2)
hold on
plot3(marbleTrajectory(:,1), marbleTrajectory(:,2), marbleTrajectory(:,3), 'r*')
plot3(marbleTrajectory(20,1), marbleTrajectory(20,2), marbleTrajectory(20,3),
'db', 'MarkerSize', 12)
% -5.6250    4.2188   -6.0938
axis([-12 12 -12 12 -12 12]), grid on
hold off;
```

C References

https://www.math24.net/fourier-series-definition-typical-examples/
MATH 482 Course Notes