AMATH482 HOMEWORK2

CIHAN MERTOZ

Abstract. Signal processing is one of the most important developments in the recent history of mathematics. In the last homework, the Fast Fourier Transform was used to analyze a data sets. In this Homework, unlike the last one, our signal is not static which means it changes over time. Since the Fourier Transform only gives information about the the frequency domain, we need use another technique to get information about both the time and the frequency of signal while still obeying the Heisenberg Uncertainty Principle

1. Introduction and Overview

Since there is data containing information about the location of the marble, the data needs to be analyzed with proper tools. However there are couple of challenges to analyze the data. Firstly, data is very noisy which means there is unwanted signal that need to be eliminated. This arises because the dog is moving which makes the fluids inside the intestines to move. Lastly, we need to find the exact location of the marble at the 20th point so that we can direct our intense acoustic wave to break the marble and save the dog

2. Theoretical Background

2.1. **Fourier Transform.** The approach to this problem is using the Fast Fourier technique developed by the French mathematician Joseph Fourier. Fast Fourier technique, shortly fft, is a major development in the 20th century for signal processing problems. The idea behind Fourier Transforms is simple. We can represent any function with a set of sine and cosine functions which is called the Fourier series.

$$f(x) = \frac{a_0}{2L} + \sum_{n=1}^{\infty} a_n cos\frac{\pi nx}{L} + b_n sin\frac{\pi nx}{L}$$

When L is at the infinity, the Fourier Series become the widely known Fourier Transforms

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

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

In this problem, the fft tool developed by MATLAB is used to tackle the problem. fft in MATLAB applies the same principles of Fourier Transforms to many data points efficiently, the operation count is O(NlogN)

Date: February 2020.

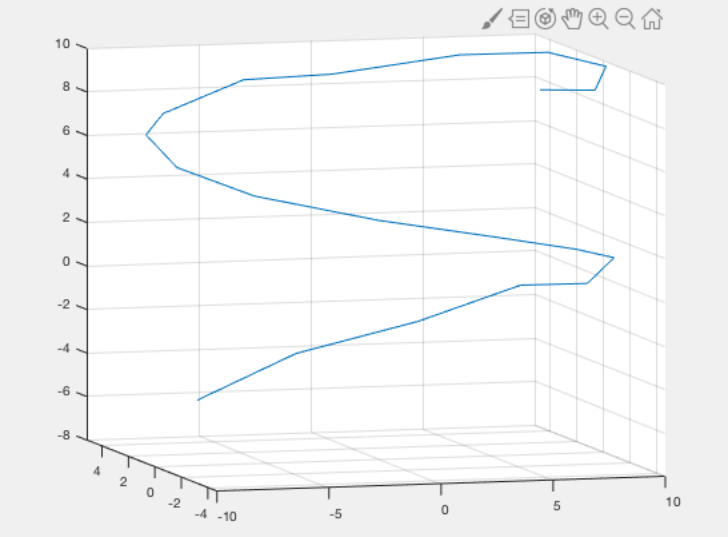
2.2. **Gaussian Filtering.** Analogous to a dirty water that needs filtration, our data also needs to be filtrated to determine the path of the marble in the intestines of the dog. To do that a tool called Gaussian filter is used.

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

where τ measures the bandwidth of the filter, and k is the wave-number.

3. Algorithm Implementation and Development

Having a noisy data set demands the data set to be cleaned.To do that the average of the frequencies was found by adding every entry of the data set together and dividing it by the size of the data set. To eliminate data redundancy, data is normalized. Since there is signature frequency emitted by the marble, that needs to be found in order to trace that frequency in the intestines. The signature frequency is found by tracing the maximum signal detected. This makes sense assuming the noise is white and similar all around the intestines. If there is an object passing by the frequency should be automatically increased. Then a Gaussian Filter is built to filter the data using the signature frequency. After many errors, the highest possible bandwidth value is chosen. Lastly the filter function is applied over a Fourier transform and data points corresponding to the path of the marble are appended to a matrix to plot the pathway of the marble.



4. Computational Results

4.1. **Center Frequency.** Center Frequency is at (1.8850,-1.0472,0) as calculated in line 34 of Appendix B.

4.2. **Path of the Marble.** The path of the Marble can be seen in figure generated in line 58 of Appendix B.

4.3. **Last Stop of the Marble.** The 20th data measurement, which is the last stop of the marble in intestines, is at(-5.6250,4.2188,-6.0938).

5. Conclusion

In this problem we used one of the most important signal processing tools available to find the path of a marble stuck in our dogs intestines. After finding the average frequency we applied a filter using that frequency to get rid of noise and then find the path of the marble.

6. Appendix A

```
clear; close all;
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);


% Define U and Usize , it will be used in both of the for loops
U = Undata;
Usize = size(U,1);


%Initiliaze an empty vector to add in the for loop
sum = zeros(n,n,n);

for j= 1:Usize
    u(:,:,:) = reshape(U(j,:),n,n,n);

    ut = fftn(u);
    % Append all the frequency data to one vector
    sum= sum+ut;
end


% Find the average and normalize
uave = abs(fftshift(sum))/Usize;
uaven = uave/max(uave(:));
ind2 = find(uaven ==1);

% Find the central frequency in each direction by using
kx = Kx(ind2);ky = Ky(ind2);kz = Kz(ind2);

%Define a bandwidth, I played around with many values and found the highest
%optimized bandwith value
bandwith = 4.56;

%Gaussian Filter
filter = fftshift(exp(-1*bandwith * ((Kx-kx).^2 + (Ky-ky).^2 + (Kz-kz).^2)));
%empty 20x3 matrix to append the path of the marble
marble_path = zeros(20,3);
```

```
for times = 1:Usize
    % Find x, y, z coordinates
    u(:,:,:)=reshape(U(times,:),n,n,n);
    unf = fftshift(fftshift(ifftn(fftn(u) .* filter)));
    [A, B] = max(unf(:));
    [X_path, Y_path, Z_path] = ind2sub([n,n,n], B);
    %marble path appended
    marble_path(times, 1) = X(X_path, Y_path, Z_path);
    marble_path(times, 2) = Y(X_path, Y_path, Z_path);
    marble_path(times, 3) = Z(X_path, Y_path, Z_path);

end

plot3(marble_path(:,1),marble_path(:,2),marble_path(:,3)); grid on

%where the marble is right now
last_stop = marble_path(20,:)
```

7. Appendix B,Functions Used

fftn(x) = computes the discrete Fourier transform (DFT) of X using a fast Fourier transform (FFT) algorithm.

fftshift(X) rearranges a Fourier transform X by shifting the zero-frequency component to the center of the array.

[I1,I2,...,In] = ind2sub(sz,ind) returns n arrays I1,I2,...,In containing the equivalent multidimensional subscripts corresponding to the linear indices ind for a multidimensional array of size sz. Here sz is a vector with n elements that specifies the size of each array dimension.