Homework 1: An Ultrasound Problem

Maximilian Walter, AMATH 482, University of Washington

This assignment uses frequency analysis and other tools to process noisy 3D spatial data into usable information. By reducing noise and examining the data in frequency space, a hard object's distinct frequency signature can be determined from ultrasound data. From this information, a filtering function can be used to denoise the data and determine the spatial location of the object traveling through the body.

Introduction and Overview

My dog, Fluffy, accidentally ate one of my marbles from my exquisite collection. At the veterinarian's office, an ultrasound was done to locate the marble. Fluffy refuses to hold still, and thus, the fluid in the intestine is causing a lot of noise within our signal. We can take 20 distinct time points resulting in a data set that is [20 x 64 x 64 x 64]. I am determined to use this data to find the marble and save my dog.

Theoretical Background

Fourier Series and Transform

The Fourier series and transform are analysis tools to help us better understand a signals process. The Fourier series is a mathematical equation that allows the user to recreate a signal by combining a series of Sines and Cosines at different frequencies. The Fourier Transform is a tool to analyze an incoming signal by decomposing it into the frequencies that make up that signal. Similar to music, each chord has a series of notes, that together make up the sound we hear. The Fourier transform allows the user to pick out the individual notes and their strengths, which in this case is frequency. The transform relies heavily on Euler's formula, which allows the function to have complex exponential values, which we can represent with sines and cosines.

Euler's Formula

$$e^{i\theta} = cos\theta + isin\theta$$

This formula allows us to take the transform using the following equation.

Fourier Transform

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

And to return from frequency to spatial domain the inverse Fourier Transform is required.

Inverse Fourier Transform

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

With these components, we are able to break down our original noisy signal and look for specific frequencies that could be the marble we are looking for.

Gaussian Function

To build the filter, a Gaussian curve is used as the base function which is manipulated to contain the targeted frequency. Kx,y,z are the grid locations and Fx,y,z are the desired center frequency. Tau allows me to alter the width of the curve to include a wider range of surrounding frequencies.

Gaussian Filter

$$e^{-\tau[(K_x-F_x)^2-(K_y-F_y)^2-(K_z-F_z)^2]}$$

Algorithm Implementation and Development

The process of finding the marble involves denoising the data, determining the marbles frequency signature, and filter the original signal based on this signature to locate the marble.

I begin by creating the boundaries of my spatial and frequency domain. These were preset by the instructor. We are creating a mesh grid of 64x64x64 points between the values of [-15,15]. I also rescale my frequencies based on this information to maintain a 2*pi periodic signal for work done using Fourier Transforms and in the frequency domain. The signal is also shifted to better align around the center axis using the command fftshift.

Denoising the Signal (Lines 19-26)

The first step in denoising the signal was to use averaging as a way to reduce the noise and help the marbles frequency to become more prominent. This is done by summing each timepoints Fourier transform data and dividing by the number of time points (20). The final result is also shifted, to match to correct frequency region as our k values. This reduces most white noise as they should average to zero over time. This allowed me to find the maximum value of the signal, which I determined to be the signal from the marble. Using the index of the maximum signal, I was able to locate it within the frequency space mesh grid.

Filtering the Signal (Lines 27-36)

After locating the signal in frequency space, I used the center frequency to create a gaussian filter. The values are of Kx,y,z are the grids frequencies, with Fx,y,z being the center frequency.

$$filter = e^{-0.2[(K_x-F_x)^2-(K_y-F_y)^2-(K_z-F_z)^2]}$$

I can then multiply the shifted original signal in frequency space by this filter to remove unwanted noise and single out my desired frequency. I can then return to the spatial domain and store this as a new set of data.

Locating the Marble (Lines 37-58)

Like before, I determined that the maximum signal should be my marble after denoising. I use the same indexing tool to locate the marbles index value and then convert that into the subscripts of my spatial mesh grid. Plotting this location in 3D, allows me to see the trajectory of my marble over the 20-time points. Using the location information at the 20th time point, I can use this as a target for my ultrasonic pulse and destroy the marble.

Computational Results

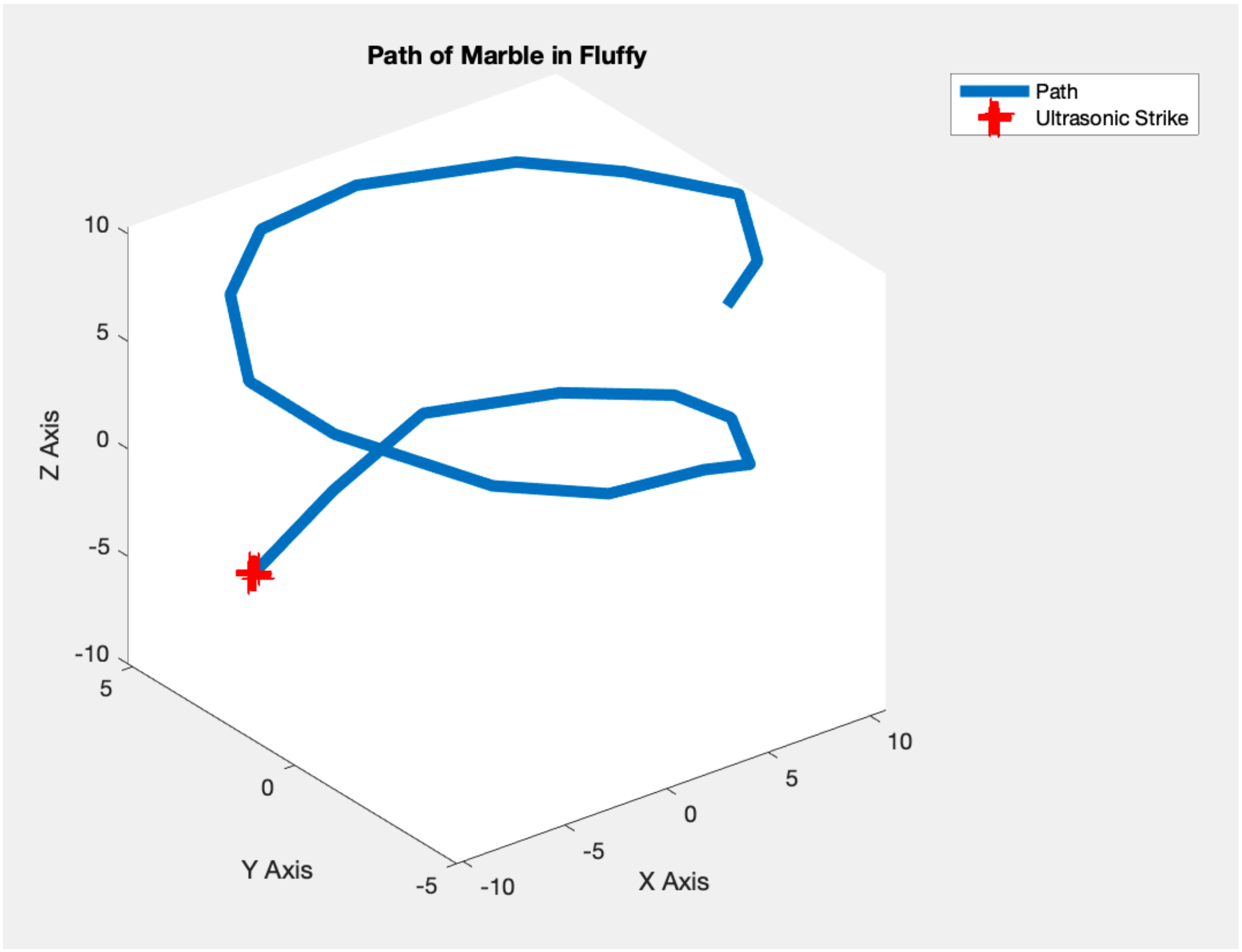

Figure 1. This is the trajectory of the marble traveling through fluffy. The red X signifies the location at t = 20 and where the ultrasonic pulse should be targeted, [-5.6520, 4.2188, -6.5625], to destroy the marble.

Summary and Conclusions

After reducing white noise from my original signal using averaging, I was able to determine the marbles distinct frequency signature. Using this signature, I was able to create a Gaussian filter, centered around this frequency and filter my original signal. Using the final signal and returning to the spatial domain, I could accurately plot the marbles trajectory as seen in Figure 1. The final location of the marble at the 20th time point was [-5.6520, 4.2188, -6.5625]. I could then use an ultrasonic pulse to break apart the marble. In reality, I would not recommend this method, and instead, allow the marble to pass naturally, as shards of glass in the intestines are not recommended.

Appendix A.

Meshgrid(x,y,z): This allows me to defined a 3D region filled with coordinates. The boundaries are defined by x, y, z values. This will allow me to locate objects within this space and have a "working" region for my functions and commands.

Fftshift(k): Shifts the zero frequencies of a signal to the middle. This allows the user to see more data centered around the 0-frequency axis.

Reshape(X, l, m, n): This function takes column-wise entries from a data set X and places them into a defined space. In our case, we place our data that comes out as a [20x262144] and places them into a 64x64x64 matrix.

Fftn(x): This is a n-dimensional discrete Fourier transform function. This allows me to take the Fourier transform of my higher dimensional signal.

Ind2Sub(SIZ, IND): This function allows me to take the index of my signal, in this case, the maximum signal, and translate that into the subscript of a 64x64x64 matrix. This allows me to quickly find the location of my signal within my data.

Appendix B.

```
%Max Walter
%AMATH 482
%Homework 1
%1/15/2020

clear all; close all; clc;

load('Testdata.mat')

L=15; % spatial domain
n=64; % Fourier modes

%Space
x2=linspace(-L,L,n+1);
```

```
x=x2(1:n);
y=x;
z=x;
[X,Y,Z]=meshgrid(x,y,z); %64x64x64 Grid for x,y,z spatial location

%Frequency
k=(2*pi/(2*L))*[0:(n/2-1) -n/2:-1];
ks=fftshift(k);
[Kx,Ky,Kz]=meshgrid(ks,ks,ks); %Grid of Frequency from -2pi to 2pi

%Using Avg. to cancel/reduce whitenoise
fft_sum = zeros(64,64,64);
for j=1:20
Un(:,:,:)=reshape(Undata(j,:),n,n,n);
fft_run = fftn(Un);
fft_sum = fft_sum + fft_run;
end
abs_fft_sum = abs(fftshift(fft_sum/20));

%Finding maximum value and index in avg. data
[value,index] = max(abs_fft_sum(:));

%using index to find subscripts
[sX,sY,sZ] = ind2sub([n,n,n],index);

%Converting to frequency
Fx = Kx(sX,sY,sZ);
Fy = Ky(sX,sY,sZ);
Fz = Kz(sX,sY,sZ);

%Creating Filter using Gaussian Filter from book
filter = exp(-0.2 * ((Kx - Fx).^2 + (Ky - Fy).^2 + (Kz - Fz).^2));

%Initalizing Positions
x_pos = zeros(1,20);
y_pos = zeros(1,20);
z_pos = zeros(1,20);

%Finding Marble Position
for j=1:20

%Filtering Signal
Un(:,:,:)=reshape(Undata(j,:),n,n,n);
fft_run = fftn(Un);
filtered_signal = filter .* fftshift(fft_run);

%Returning to Spatial
Clean_Position = real(ifftn(filtered_signal));

%Finding Marble using largest signal like in previous method
[value,index] = max(Clean_Position(:));
[sX,sY,sZ] = ind2sub([n,n,n],index);

%Storing Location
x_pos(j) = X(sX,sY,sZ);
y_pos(j) = Y(sX,sY,sZ);
```

```
z_pos(j) = Z(sX,sY,sZ);
end

%Ultra-sonic strike
endpoint = [x_pos(end), y_pos(end), z_pos(end)];

figure(1)
plot3(x_pos,y_pos,z_pos, 'Linewidth', 5)
hold on
plot3(x_pos(end), y_pos(end), z_pos(end),'ro', 'Linewidth', 10)
xlabel('X Axis'); ylabel('Y Axis'); zlabel('Z Axis')
legend('Path','Ultrasonic Strike'); title('Path of Marble in Fluffy')
```