Homework 1

Ofek Inbar

Abstract Given a noisy three-dimensional ultrasound signal, can we find the marble and track it over time? Utilizing Fast Fourier Transforms along with the averaging and filtering techniques for de-noising a signal, we found success in determining the exact position of the marble over the 20 time slices of data we worked with.

1 Introduction and Overview

We are given noisy data from an ultrasound of a dog's intestines, where a marble is lodged. In order to save the dog, we must determine the location of the marble amid the noisy ultrasound data, so as to discover the location towards which can focus an intense acoustic wave to break it up.

2 Theoretical Background

We use two methods of de-noising in this paper, averaging and filtering. Both of these methods rely on it being easy and fast to transform between a raw signal and its Fourier Transform. The Fourier Transform of a signal is a function that provides the values for the coefficients of its Fourier Series, a sum of sines and cosines of varying frequencies.

Averaging refers to taking the average of each value of the Fourier Transform at each point in time. Since "noise" is expected to contribute an average of 0 to each frequency (for large n), we are able to cancel out discrepancies caused by noise, making it possible to distinguish the marble's three-dimensional "frequency" from the rest of the signal.

Filtering refers to the amplification of frequencies close to the base frequency (assuming we already know it) and diminishing those frequencies that are farther away. When we perform an inverse Fourier transform on the filtered frequencies, we get back the original signal, but the values associated with random noise are greatly diminished, to the point where we can easily identify the marble among the rest.

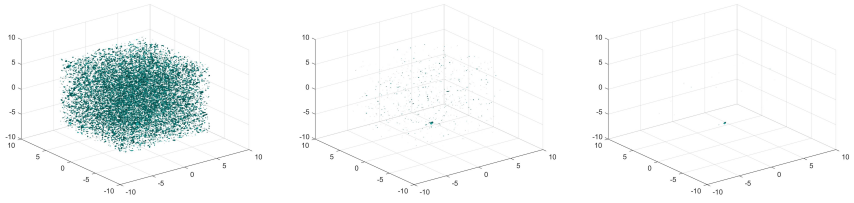

Figure 1: Averaged signal with isosurface values 100, 150, 190 (left to right)

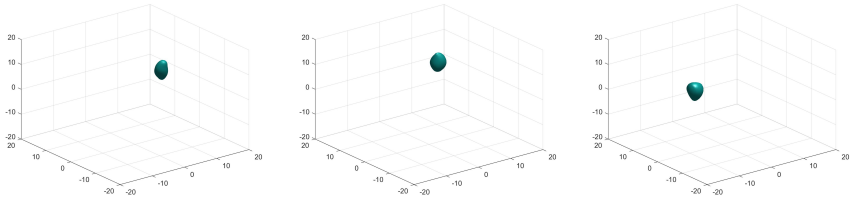

Figure 2: The de-noised signals at time $t = 1, 5, 10$ (left to right)

3 Algorithm Implementation and Development

The first step was to generate an averaged frequency signal from the transforms of each of the 20 time slices. After performing a Fast Fourier Transform on each slice, summing them at each point, and dividing the whole thing by 20, we are left with a signal that has a clear maximum, as is shown in Figure 1.

Next, we extracted the x, y, and z components of this frequency. Using these, we were able to build a Gaussian filter in the frequency space. We applied this filter to the transform signal of each time slice (by multiplying them together) before inverting the transform to get back the original, de-noised image, as can be seen in Figure 2.

4 Computational Results

We found that the averaging and filtering techniques were very effective at de-noising the three-dimensional ultrasound signal. By utilizing these methods we were able to determine with a high level of confidence the location of the marble at each time slice. Please refer to Figure 3 to see the trajectory of the marble in the ultrasound.

We found that the co-ordinates of the marble at slice 20 was

$$(-5.1563, 4.2188, -6.0938).$$

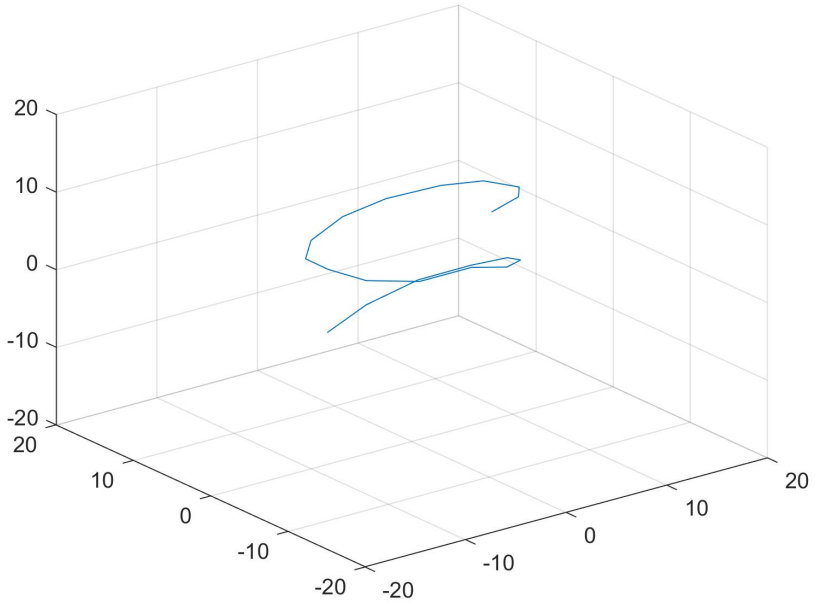

Figure 3: The path of the marble throughout the 20 time slices

5 Summary and Conclusions

The averaging and filtering techniques used in this study were found to be very effective at removing noise from a very noisy three-dimensional signal. By utilizing these techniques we were able to precisely determine the location of the marble at every time slice, even though it was moving around. We thus conclude that the Fast Fourier Transform, alongside such techniques as mentioned above, are well-suited to handle problems such as we have addressed today. We also conclude that fluffy will survive his operation.

6 Appendix A

We used the MATLAB functions `fft`, `ifft`, `fftshift`, and `ifftshift` to go back and forth between the original signal and its Fourier Transform. The first two functions perform the FFT or inverse FFT, while the latter two shift the data so the 0 frequency is centered (or invert that shift).

The first section of MATLAB code consists of the starter code provided, followed by a group that averages the Fourier Transforms of the 20 time slices, a group that plots the resulting frequencies, and a group that finds the frequency of the marble.

The second section defines the filter around the frequency found at the end of the previous section, and then defines a function to apply that filter. It then plots the result of filtering the Fourier Transform of one of the time slices and then inverting the transformation to get back a de-noised signal. Finally, it tracks the trajectory of the marble through each slice and plots it.

The final section just displays the co-ordinates of the marble in the final, 20th time slice.

7 Appendix B

```
%% Find center frequency
clear; 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);

Utave = zeros(1, n^3);
for j = 1:20
    Utave = Utave + fft(Undata(j, :));
end
Utave = Utave / 20;

isosurface(X, Y, Z, abs(reshape(fftshift(Utave),n,n,n)), 190);
axis([-20 20 -20 20 -20 20]), grid on;

[Utmax, Utmaxi] = max(abs(fftshift(Utave)));
center = [Kx(Utmaxi), Ky(Utmaxi), Kz(Utmaxi)];

%% Find marble trajectory
tau = 0.5;
filter = reshape(exp(-tau * ((Kx-center(1)).^2 ...
    + (Ky-center(2)).^2 ...
    + (Kz-center(3)).^2)),1,n^3);
f = @(j) filter .* fftshift(fft(Undata(j, :))); % apply filter

isosurface(X, Y, Z, abs(reshape(ifft(ifftshift(f(10))),n,n,n)), 0.1);
axis([-20 20 -20 20 -20 20]), grid on;

x = zeros(1, 20); y=x; z=x;
```

```
for i=1:20
    [m, idx] = max(abs(ifft(ifftshift(f(i)))));
    x(i) = X(idx); y(i) = Y(idx); z(i) = Z(idx);
end

plot3(x, y, z);
axis([-20 20 -20 20 -20 20]), grid on;

%% Find location of marble at t=20
[x(20), y(20), z(20)]
```