An Ultrasound Problem

University of Washington
AMATH 582 Homework 1
Winter Quarter 2019

David Long
dmlong888@live.com

Abstract

The purpose of this report is to explore applications of the Fast-Fourier Transform to sonography. Simulated noisy ultrasound data was transformed in order to locate the frequency signature of a marble in a dog's small intestine. The path of the marble was then charted and a trap was set to intercept the marble and annihilate it with an intense acoustic blast.

I. Introduction and Overview

The setting for this investigation is an ultrasound problem. A fluffy dog has swallowed a marble, which has worked its way into her intestines. The object is to track the marble over time, chart its trajectory, then blast it and destroy it with a super-intense acoustic wave, saving not just the fluffy dog, but at least 0.001% of the universe as well.

The data collected by the ultrasound device are noisy and must be filtered to isolate the marble. This calculation will be accomplished by transforming the data to a different basis which facilitates the filtering process.

Han and Chewie will flush the marble your way with the Falcon. Have your X-wing on standby.

II. Theoretical Background

The pioneer of sonography is credited as being physiologist Lazzaro Spallanzani, who studied echolocation among bats. Modern ultrasound, however, really exists because of the ground-breaking work of Pierre and Jacques Currie, who discovered piezoelectricity in 1877[2]. The brothers discovered that when pressure is applied to quartz or similar crystals, an electrical charge is created. By sandwiching the crystal between two plates of metal and applying external pressure, a voltage potential is created. Microphones are an example of technology that uses the piezoelectric effect.

Ultrasound exploits the inverse piezoelectric effect[1]. By applying a current to the sandwiched crystal, a transducer is created which is capable of producing soundwaves that are inaudible to humans. The same transducer collects data about the reflected soundwaves. The ultrasound waves are typically in the range of 20 kHz to several GHz[3]. By filtering the data from the reflected soundwaves, doctors are able to non-invasively investigate their patients.

The filtering of the data in this investigation will be accomplished with the Fast-Fourier Transform (FFT). The FFT moves a dataset from a time basis to a frequency basis, whereby frequencies of high intensity can be identified and isolated. Applying the inverse FFT to the filtered frequency data brings it back to the time basis, bringing with it knowledge of the source of the isolated frequency.

The Fast-Fourier Transform algorithm was developed by Cooley and Tukey in the mid-1960s. It's efficiency of $O(N \log N)$, where N is the number of points to be transformed, is the reason that it is considered one of the most important algorithms of all time[4]. It makes transformations of tremendously large data sets computationally tractable. The equations to transform from time t to frequency k and to invert are given by

$$F(k) = \frac{1}{\sqrt{2\pi}} \int_{-\infty}^{\infty} e^{-ikt} f(t)\, dt \qquad \text{(Eqn. 1.1.a)}$$

$$f(t) = \frac{1}{\sqrt{2\pi}} \int_{-\infty}^{\infty} e^{ikt} F(k)\, dk \qquad \text{(Eqn. 1.1.b)}$$

In our setting, the spatial domain is confined to the dog's intestines. The reflected frequency signature of a foreign object such as a marble will differ considerably from that of dog guts. This signature will be isolated in the Fourier domain and used as a center for filtering, after which we will be able to track the marble in the spatial domain.

To isolate the marble's frequency signature, data must be collected over several slices of time. In the Fourier spectrum, white noise is modeled by a randomly distributed normal variable with mean of zero and variance of one [4]. The implication of this is that the sum of the white noise of repeated measurements will approach zero in the long run, offering a clearer view of the frequency spectrum in the process. Here, the frequency signature of the marble can be detected. Filtering out the other frequencies effectively isolates the marble from the intestines.

The filtering is accomplished by multiplying the data in the Fourier domain by a Gaussian function, which has the effect of preserving the frequency spectrum near the center of the filter and zeroing out everything that is far away. The intestinal domain is three-dimensional, so the Gaussian will be of the form

$$\Gamma\left(k_x, k_y, k_z\right) = \exp\left\{-\tau \bullet \left[\left(k_x - k_{x_0}\right)^2 + \left(k_y - k_{y_0}\right)^2 + \left(k_z - k_{z_0}\right)^2\right]\right\}, \qquad \text{(Eqn. 1.2)}$$

where τ controls the spread around the center $\left(k_{x_0}, k_{y_0}, k_{z_0}\right)$. The inverse Fourier transform of the filtered frequency data will reveal the path of the marble in space.

III. Algorithm Implementation and Development

Though the Fourier expansion of $f(t)$ is integrated over all real numbers, the computational domain must be finite. As the number of Fourier modes was given to be $n = 64$, the domains of the *x*-, *y*-, and *z*- axes were bounded at $\pm L = \pm 15$ and discretized into $n+1$ points. To account for the periodicity of the basis, the boundary values must be equal, so only the first *n* points are considered in each direction. A vector space in R^3 was then constructed with the `meshgrid` command.

The Fourier space was similarly constructed. There were $n = 64$ wavenumbers, which had to be scaled to from a length of $2L$ to a length of 2π. To comply with the FFT algorithm, the scaled domain needs to be shifted so that the non-negative wavenumbers precede the negative wavenumbers. All of this was accomplished with the command

```
k = (2*pi/(2*L))*[0:(n/2-1) (-n/2):-1];
```

For purposes of visualization, the scaled wavenumbers were inverse shifted and meshed together in three dimensions.

The data were loaded into MATLAB as twenty rows of three-dimensional data collected at twenty points in time. For both visualization and computation, the data had to be reshaped into $64x64x64$ matrices. To organize the twenty realizations, I constructed a $20x1$ cell array and populated it with the reshaped matrices. This vector stored the noisy time data. I reserved space for similar arrays to hold the transformed data, the filtered transformed data, and the filtered data. The use of cell arrays simplified my code and thought process compared to working with four-dimensional matrices.

The first step in identifying where to center the filter in the Fourier domain was to average out the white noise. This was accomplished with a for-loop over time that had just two statements, one to transform the noisy data with `fftn()` and another to accumulate the sum of the data. Of concern in the transformed data is the scaled intensity, so the transformed data were divided by the absolute value of the maximum. The shifted absolute values were plotted against the Fourier meshgrid with `isosurface()` **(Figure 1)**. I kept upping the threshold towards one and let MATLAB choose the boundaries for visualization. It finally zoomed in on a surface that had four vertices with very similar coordinates, so I centered the filter in that region **(Figure 2)**.

The Gaussian filter was centered at $\left(k_{x_0}, k_{y_0}, k_{z_0}\right) = (1.9, -1, 0)$ and is governed by (Eqn. 1.2):

$$\Gamma\left(k_x, k_y, k_z\right) = \exp\left\{-\tau \bullet \left[\left(k_x - 1.9\right)^2 + \left(k_y + 1\right)^2 + \left(k_z - 0\right)^2\right]\right\}$$

I experimented with various values of τ, but found that $\tau = 1$ worked just fine.

I used another for-loop to filter the transformed data by multiplying it component wise with the shifted and transformed noisy data matrix and stored it, then inverse shifted the result and applied `ifftn()` to bring the filtered and transformed data back into the spatial domain.

Having stored everything in cell arrays made it really easy to visualize the filtered data. I was able to simply loop through the array and add the next time realization to the same `isosurface()` plot. For both domains, I set the axes to the entire computational domain. In the frequency domain, it looked like successive spherical scatterplots centered at $\left(k_{x_0}, k_{y_0}, k_{z_0}\right) = (1.9, -1, 0)$ **(Figure 3)**. In the spatial domain, it looked like a marble moving through an intestinal tract, or at least a helix-shaped simulation of one **(Figure 4)**.

I considered using the maximum frequency to locate the center of the marble, but it occurred to me that due to the white noise, the maximum frequency would not necessarily be at the center. Instead, I exported the vertex structures of the `isosurface()` and calculated the centroids at each point in time. The final time realization is shown in **(Figure 5)**. The centroids were used to plot the trajectory of the marble and to target the marble with a destructive acoustic blast **(Figure 6)**.

IV. Computational Results

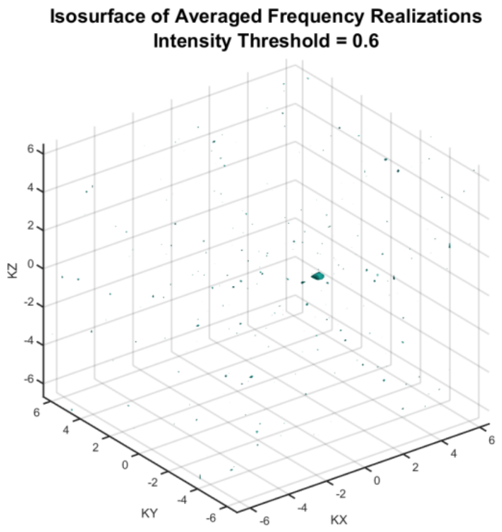

Figure 1: After twenty time realizations, much of the white noise has been averaged out. With the intensity threshold set at 0.6, a relatively large frequency region is identified.

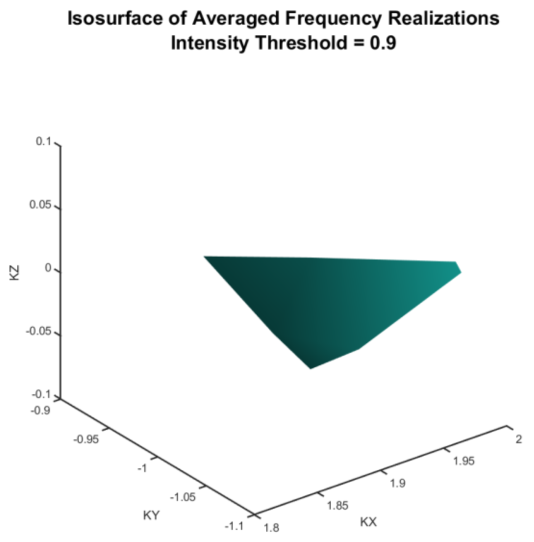

Figure 2: Successively increasing the intensity threshold consistently reveals a strong frequency signature centered at $\left(k_{x_0}, k_{y_0}, k_{z_0}\right) = (1.9, -1, 0)$.

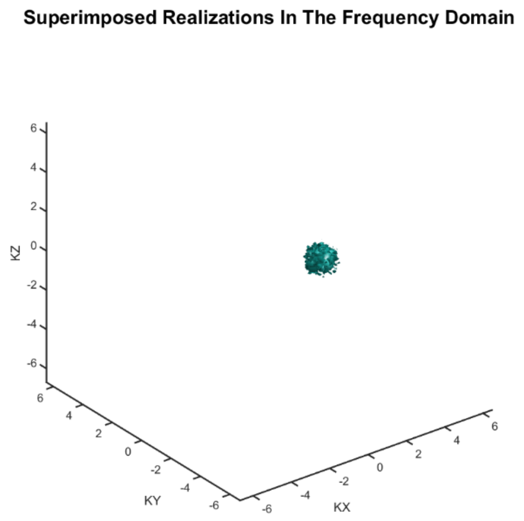

Figure 3: After multiplying by the Gaussian $\exp\left\{-\left[\left(k_x - 1.9\right)^2 + \left(k_y + 1\right)^2 + \left(k_z - 0\right)^2\right]\right\}$, the filtered data were superimposed over time. Each realization was a consistently sized cluster in the frequency domain.

the isosurface() seem to accurately pinpoint the enemy's location. Time to launch the X-wings.

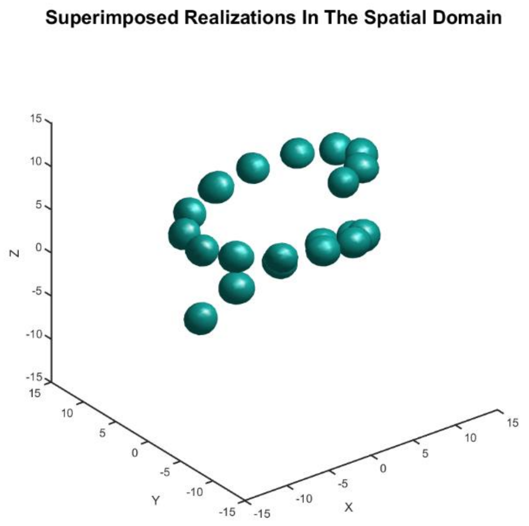

Figure 4: After transforming the filtered frequency data back to the spatial domain, the path of the marble is revealed.

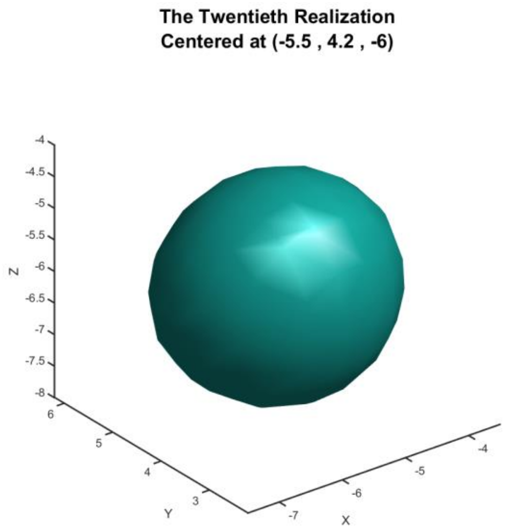

Figure 5: Where the rogue marble can be found right now. The coordinates of the centroid of the `isosurface()` seem to accurately pinpoint the enemy's location. Time to launch the X-wings.

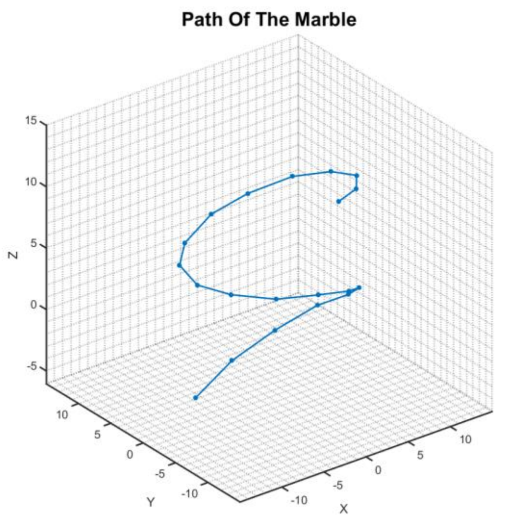

Figure 6: MATLAB returned $(-5.5070, 4.2049, -6.0544)$ as the current coordinates of the rogue marble. Gut check. Who do you trust, MATLAB or the force? Or are they one and the same?

V. Summary and Conclusions

Visualization of the simulated noisy ultrasound data was nonsensical. White noise buried the location of the marble. The power of the Fourier transform is on full display here, as it allows for the data to be represented in a different basis, where it can be easily manipulated and sent back to the original basis, revealing what was previously obscured. We should also give a nod to the Central Limit Theorem, who very quietly filtered out the white noise for us.

Readers will be delighted to learn that the fluffy dog did indeed survive. The acoustic blast was delivered just at the last moment, but the pilot was shot down by a tie fighter. We're trying to recover the black box, but at this point, we don't know whether he used MATLAB or the force.

A. MATLAB Functions

B. MATLAB Codes

```
% Homework 1 - An Ultrasound Problem
% David Long
% 1.12.19

clear all; close all; clc


% Computational domain

L = 15;
n = 64;
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);




% Load data

load Testdata
slice = transpose(1:size(Undata, 1));
Un = cell(length(slice), 1);
for j=1:length(slice)
    Un{j} = reshape(Undata(j,:), n, n, n);
end

Unt = cell(length(slice), 1);
Untf = cell(length(slice), 1);
Unf = cell(length(slice), 1);




% Determine frequency signature through averaging

UntAvg = zeros(n, n, n);

for j=1:length(slice)
    Unt{j} = fftn(Un{j});
    UntAvg = UntAvg + Unt{j};
end
```

```
% Plot average frequency

figure(1)
isosurface(KX, KY, KZ, fftshift(abs(UntAvg))/max(max(max(abs(UntAvg)))), 0.6)
set(gca, 'Fontsize', 14, 'LineWidth', 2)
grid on
axis([ks(1) ks(end) ks(1) ks(end) ks(1) ks(end)])
xlabel('KX'), ylabel('KY'), zlabel('KZ')
title({'Isosurface of Averaged Frequency Realizations';
        'Intensity Threshold = 0.6'}, 'Fontsize', 24)

figure(2)
isosurface(KX, KY, KZ, fftshift(abs(UntAvg))/max(max(max(abs(UntAvg)))), 0.9)
set(gca, 'Fontsize', 14, 'LineWidth', 2)
grid off
axis([1.8 2.0 -1.1 -0.9 -0.1 0.1])
xlabel('KX'), ylabel('KY'), zlabel('KZ')
title({'Isosurface of Averaged Frequency Realizations';
        'Intensity Threshold = 0.9'}, 'Fontsize', 24)
```

```
% Filter around center frequency

filter = exp(-1 * ((KX - 1.9).^2 + (KY + 1).^2 +(KZ - 0).^2));

for j=1:length(slice)
    Untf{j} = filter.*fftshift(Unt{j});
    Unf{j} = ifftn(ifftshift(Untf{j}));
end
```

```
% Plot filtered frequency data

for j=1:length(slice)
    figure(3)
    isosurface(KX, KY, KZ, abs(Untf{j})/max(max(max(abs(Untf{j})))), 0.5);
    axis([ks(1) ks(end) ks(1) ks(end) ks(1) ks(end)])
end
set(gca, 'Fontsize', 14, 'LineWidth', 2)
grid off
xlabel('KX'), ylabel('KY'), zlabel('KZ')
title('Superimposed Realizations In The Frequency Domain', 'Fontsize', 24)
```

```
% Plot filtered spatial data

for j=1:length(slice)
    figure(4)
    isosurface(X, Y, Z, abs(Unf{j})/max(max(max(abs(Unf{j})))), 0.7);
    axis([-L, L, -L, L, -L, L])
end
set(gca, 'Fontsize', 14, 'LineWidth', 2)
grid off
xlabel('X'), ylabel('Y'), zlabel('Z')
title('Superimposed Realizations In The Spatial Domain', 'Fontsize', 24)




% Estimate centers of marble

center = cell(length(slice), 1);

for j=1:length(slice)

    [f, v] = isosurface(X, Y, Z, abs(Unf{j})/max(max(max(abs(Unf{j})))),
0.7);
    center{j} = mean(v);

end

for j=1:length(slice)
    currentCenter = center{j};
    xcenter(j) = currentCenter(1);
    ycenter(j) = currentCenter(2);
    zcenter(j) = currentCenter(3);
end




% Center final realization in space

figure(5)
isosurface(X, Y, Z, abs(Unf{j})/max(max(max(abs(Unf{j})))), 0.7)
set(gca, 'Fontsize', 14, 'LineWidth', 2)
grid off
axis([-7.5, -3.5, 2.2, 6.2, -8, -4])
xlabel('X'), ylabel('Y'), zlabel('Z')
title({'The Twentieth Realization';
        'Centered at (-5.5 , 4.2 , -6)'}, 'Fontsize', 24)
```

```
% Plot the path of the centroids

figure(6)
plot3(xcenter, ycenter, zcenter, '*-', 'LineWidth', 2)
set(gca, 'Fontsize', 14, 'LineWidth', 2, 'Xtick', [-10 -5 0 5 10], 'Ytick',
[-10 -5 0 5 10])
grid minor
axis([-L, L, -L, L, -6.054395842108247, L])
xlabel('X'), ylabel('Y'), zlabel('Z')
title('Path Of The Marble', 'Fontsize', 24)
```

C. References

[1] *What is the Piezoelectric Effect?*. (2016). *Electronic Design*. Retrieved 18 January 2019, from https://www.electronicdesign.com/power/what-piezoelectric-effect

[2] *History of Ultrasound*. (2019). *Ultrasoundschoolsinfo.com*. Retrieved 18 January 2019, from https://www.ultrasoundschoolsinfo.com/history/

[3] Wikipedia contributors. (2018, October 19). Ultrasound. In *Wikipedia*, *The Free Encyclopedia*. Retrieved 01:43, January 18, 2019, from https://en.wikipedia.org/w/index.php?title=Ultrasound&oldid=864852081

[4] Kutz, J. N. February 26, 2010. *AMATH 582: Computational Methods for Data Analysis.* Department of Applied Mathematics, University of Washington, Seattle, WA 98195.