Saving Your Dog's Life With Fourier Transformations and Signal Processing

AMATH582: Homework 1
Maple Tan

January 25, 2019

1 Abstract

Our goal is to find the trajectory of a marble swallowed by our dog. Using the Fourier Transformation, we will process noisy data obtained from an ultrasound at twenty different snapshots in time. After processing the signals, we find that the marble moves in a spiral trajectory through our dog. We also found the marble's ending location so that we may use an intense acoustic wave to break up the marble in its ending location, thus saving our dog's life.

2 Introduction and Overview

We have use of an ultrasound that returns twenty snapshots at different times that contains the position of the marble as it travels through our dog. However, due to our dog's continual movement, the returning ultrasound signals contains highly noisy data and we will need to use MATLAB extract the meaningful data buried within the noise.

3 Theoretical Background

We will extensively make use of the Fourier Transformation and the Fast Fourier Transformation algorithm. The Fourier Transformation breaks up a function into the sum of sin and cos waves and in terms of our problem, effectively finds all the different frequencies that make up our a signal. The Fast Fourier Transformation used in MATLAB is a discrete Fourier Transformation algorithm that is very efficient and have the key properties of shifting our domain and also assumes that we are on a 2π periodic.

The marble's changing location through the time shots is a major issue in this problem and makes it hard to pinpoint its signal along the noise. To help us isolate the marble's signal, we will apply the Fourier Transformation to the data, which moves the data into the frequency domain. Within the frequency domain, the physical position of the marble no longer applies and we can extract the marble's signal through averaging of our signal over the different time iterations.

Since we do not know the exact value of the marble's signal, we can add and take the average of all the data over the twenty snapshots. This averaging process will remove a good amount of the noise generated by the dog's movement and the marble will be more clearly visible in the ultrasound. In more analytic terms, this signal will be the maximum over the entire domain.

After locating the marble, we will then create a filter that will isolate the marble's signal in the frequency domain. After achieving that and applying it to each of the different time shots, we will then reverse the Fourier Transformation and the filtering in the frequency domain will carry back into the time domain and that allows us to find the location of the marble. Analytically, the marble's signal will also be the maximum over the entire time domain.

4 Algorithm Implementation and Development

The initial data is given in a 20×262144 matrix, where each row is a different time shot. At each time slot, the 1×262144 vectors each contain data of signals in a 3D space. As such, each vector must be reshaped into a $64 \times 64 \times 64$ cube. One way to understand the data is that each (x, y, z) point in the cube is a point inside our dog and the associated value from our reshaped data is the strength of the signal at that point. As seen in Figure 1, we see that our data is very noisy and nothing visually stands out in the data.

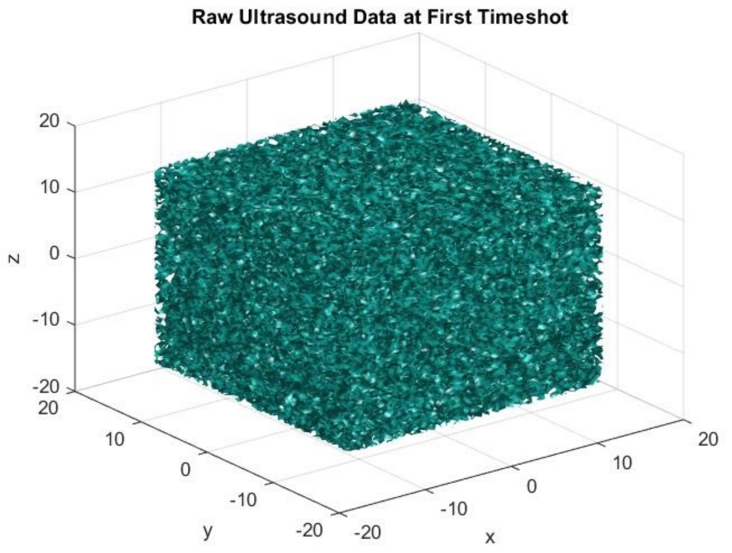

Figure 1: The visualization of our signal in the time domain at the first time shot. Note that the data is very noisy and there is no clear indication where the marble is currently located.

We have been given the spatial and spectral resolution of our ultrasound equipment will use that to create an initial x vector that matches said resolution. We also need to create our frequency domain k that matches the given spectral resolution. Since `fft` assumes 2π periodic signals and shifts our function, our frequency domain also needs to be initialized as shifted and on a 2π periodic. Finally we then create two different meshgrids using the `meshgrid` command for our time and frequency domains. For visualization purposes, we also applied `fftshift` to k before creating our frequency meshgrid. We also use `isosurface` to create our visualization.

Now we may begin with applying the Fourier Transformation to our reshaped data by using the `fftn` method. From there, we loop through the 20 time shots and sum up the signal values at each point in the cube. Taking the average over all the loops and plotting the normalized data, we obtain a much clearer image of the marble as seen in Figure 2 and Figure 3. An important thing to note is that the average must be shifted before plotting due to the nature of the `fftn` command. To keep things consistent with our visualization the next few steps involving determining the location of the marble's signal will be done with the shifted average.

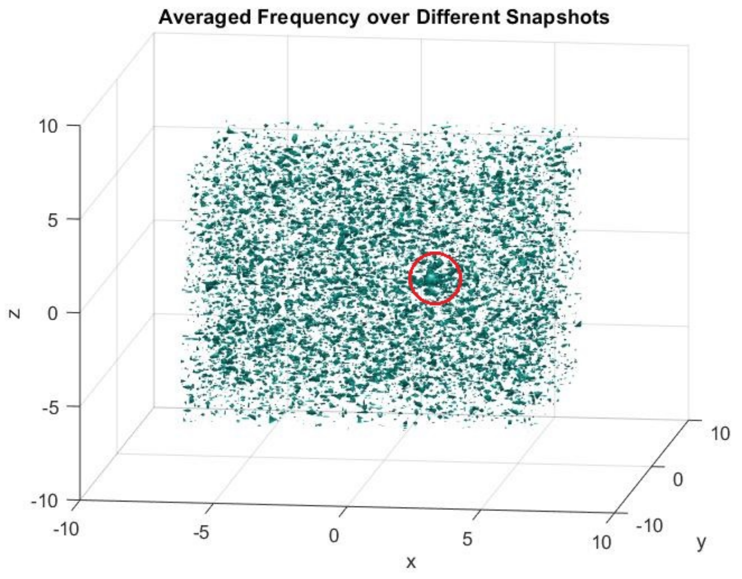

Figure 2: The visualization of our averaged signal in the frequency domain. Compared to Figure 1, there is less noise so we can see the outline of the marble as indicated by red circle. See Figure 3 for the cube in a different angle.

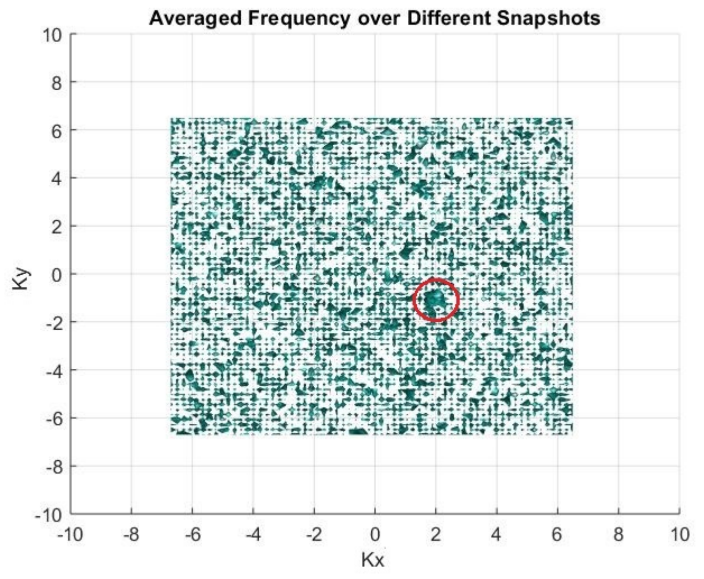

Figure 3: Our data as seen projected down to the xy plane. The marble from this angle is especially clear and indicated by the red circle.

Noise in general is normally distributed, meaning it moves towards its mean of 0 as the data is averaged. This means that the marble's signal corresponds with the maximum over all other signals. To find the value and index of the max over the entire cube, we use the `max` command which returns both the maximum value and index as if the cube was a 1×26144 vector. To find the exact indices of our maximum in our cube, we use the `ind2sub` method in MATLAB. One important thing to note is that `ind2sub` returns the indices in (y, x, z) form so the code has been adjusted accordingly to reflect that. Now that we have the indices, we take the kx, ky, and kz values at those indices to get the coordinates of our max in the frequency domain. Again, we need to remember these points have been shifted.

Now that we have the coordinates of our marble's signal in the cube, we can use that to create our filter. A basic Guassian filter has the form:

$$F(k) = exp(-\tau((k - k_0)^2)) \tag{1}$$

Which, when generalized to 3D, we have:

$$F(k) = exp(-\tau((kx - kx_0)^2 + (kx - kx_0)^2 + (kx - kz_0)^2)) \tag{2}$$

We use Equation 2 as our filter and each of the k_0's values correspond to the ks coordinate values found previously. Our τ value changes the width of our filter and just needs to be wide enough to enclose our point so a small number such as 0.2 works. Since we inputted our shifted ks values into our filter, this means our filter is shifted as well. As such, we will also have to apply the filter to the shifted version of our data as well.

Now for each of the twenty time shots of the raw data, we will apply the Fourier Transformation to the reshaped data, shift the transformed data and then apply the filter. Now when we reverse the Fourier Transformation, the image of the marble will be clear in the time domain as well. Thus allowing us to search for the maximum over all the signals in the time shot to get the index of the marble's location. Similarly the `max` function on MATLAB returns an index, so we must use that index to access the marble's actual location from our x, y and z vectors. We then store the marble location as it travels through the twenty time shots and plot the trajectory using `plot3`.

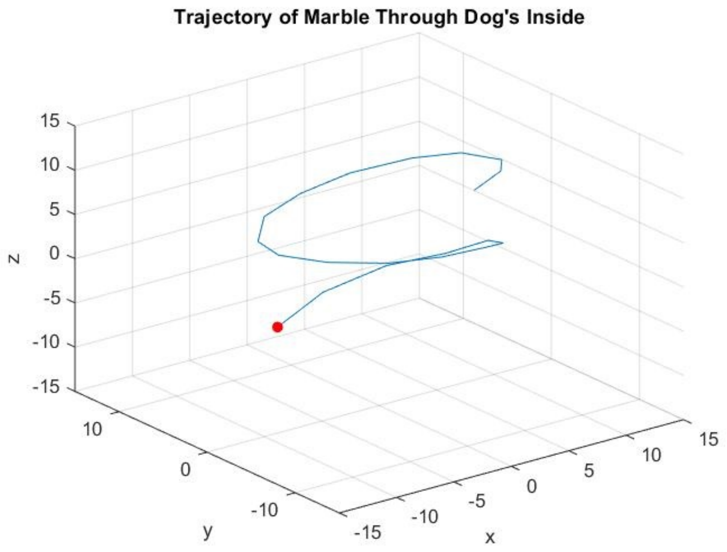

Figure 4: The trajectory of the marble through our dog's body. As we can see, the marble follows a spiral trajectory before ending at the red dot near the bottom of the graph.

5 Summary and Conclusion

Finally we have located the marble in the final time shot. We see that the marble ends up at point (-5.6250, 4.2188, -6.0938) in our cube and are able to use an intense acoustic wave to break up the marble there in the final snap shot, saving our dog's life. The importance of the Fourier Transformation is key in allowing us to extract valuable data from through seemingly unmeaningful data, especially when it comes to noisy data.

A MATLAB CODE

```
clear all; close all; clc

% Load Initial Data
load('Testdata.mat')

L=15; % spatial domain
n=64; % Fourier modes
x2=linspace(-L,L,n+1);
x=x2(1:n);
y=x;
z=x;

% Create Frequency Domain
k=(2*pi/(2*L))*[0:(n/2-1) -n/2:-1];
ks=fftshift(k);

% Create Meshgrid for both Domains
[X,Y,Z]=meshgrid(x,y,z);
[Kx,Ky,Kz]=meshgrid(ks,ks,ks);

% Plot one snapshot of the data
figure(1)
Un(:,:,:)=reshape(Undata(1,:),n,n,n);
close all, isosurface(X,Y,Z,abs(Un),0.4)
axis([-20 20 -20 20 -20 20]), grid on, drawnow
title('Raw Ultrasound Data at First Timeshot')
xlabel('x')
ylabel('y')
zlabel('z')

% Fourier Transform the Data and Take the Sum over the 20 Timeshots
Untavg = zeros(n,n,n);
for j=1:20
    Un(:,:,:)=reshape(Undata(j,:),n,n,n);
    Unf = fftn(Un);
    Untavg = Untavg + Unf;
end
```

```
% Take the Average over the 20 Time shots, Normalize the Data with
%   the Maximum and then Graph the Averaged Signal in Frequency Domain
Utavg = Untavg / 20;
Utavgshift = fftshift(Untavg) / 20;
[maxUtavg, max_ind] = max(Utavgshift(:));

figure(2)
isosurface(Kx,Ky,Kz,abs(Utavgshift)./abs(maxUtavg),0.4)
axis([-10 10 -10 10 -10 10]), grid on, drawnow
title('Averaged Frequency over Different Snapshots')
xlabel('Kx')
ylabel('Ky')
zlabel('Kz')


% Pinpoint Location of Maximum signal Strength which is our Marble's Signal
% Note that ind2sub returns in [y,x,z] form
[Ymax, Xmax, Zmax] = ind2sub([n,n,n], max_ind);
kxmax = ks(Xmax);
kymax = ks(Ymax);
kzmax = ks(Zmax);


% Use Marble Location to Create Filter
filter_eq = @(x,y,z)(exp(-0.2*((x-kxmax).^2 + (y-kymax).^2 + (z-kzmax).^2)));
filter = filter_eq(Kx,Ky,Kz);

% Apply Filter to Fourier Transformed Data in all 20 different time shots
%   before inverse transforming the data back to the time domain. Within
%   the time domain, we will locate the marble's location and track its
%   movement through the different timeshots.
X_trav = [];
Y_trav = [];
Z_trav = [];
for j=1:20
    Un(:,:,:)=reshape(Undata(j,:),n,n,n);
    Unf = fftn(Un);
    Unfs = fftshift(Unf);
    Unfs_fil = Unfs.*filter;
    Unf_fil = ifftshift(Unfs_fil);
    Un_filtered = ifftn(Unf_fil);
    [time_max, ind] = max(Un_filtered(:));
    [Ytmax,Xtmax,Ztmax] = ind2sub([n,n,n], ind);
    X_trav(j) = x(Xtmax);
    Y_trav(j) = y(Ytmax);
    Z_trav(j) = z(Ztmax);
end

% Plot Trajectory of Marble
figure(3)
hold on
view(3)
plot3(X_trav, Y_trav, Z_trav)
plot3(X_trav(end), Y_trav(end), Z_trav(end), 'r.','MarkerSize',20)
title('Trajectory of Marble Through Dog''s Inside')
xlabel('x')
ylabel('y')
zlabel('z')
axis([-15 15 -15 15 -15 15])
grid on
```