AMATH 582 Homework 1

Shabab Ahmed
Github account: sahmed95

January 24, 2020

Abstract

We find the marble in the dog's stomach by following the marble's trajectory. We use 20 different measurements taken in time about the spatial variations. We reshape the data to transform each measurement into a cube and perform time-frequency analysis on the cubes of data. However, the movement of the dog compounded with its internal movement causes the data to be highly noisy. Hence, we use the techniques of averaging and filtering to help us find the trajectory of the marble and its final position. Averaging the signal in the spectral domain we find that the center frequency is $[1.8850, -1.0472, 0]$. We filter the signal around this center frequency to find that the marble is spiraling downwards in a counterclockwise fashion. Using the trajectory, we find that the final position of the marble is [-5.6250, 4.218750, -6.093750] which is where we can focus our intense acoustic wave to breakup the marble.

1 Introduction and Overview

The dog has swallowed a marble and been taken to the vet. The vet uses ultrasound to find the position of the marble. The ultrasound gives data about the spatial variations. However, the fact that the dog and its internal fluids are moving means that this data is highly noisy. We have to find a way to denoise the data to extract the actual signal which will then be used to find the trajectory. The trajectory of the marble allows us to find the final position of the marble so that we can focus our intense acoustic wave.

Fourier analysis plays a major role in our ability to solve this problem. Fourier transform allows us to decompose a time signal into its frequency components and then derive information from it. Using certain techniques, we will be able to drown out the noise and extract the frequency signature which will then help us find the trajectory of the marble. The subject of Fourier transform is handled in more detail in the Theoretical Background section.

In what follows, we give the reader an overview. We use time-frequency analysis to find the position of the marble. To start the process, we reshape the data so that each measurement corresponds to a cube of data. We perform our analysis on these cubes and we take the average of these in the spectral domain. Since we are on a periodic domain we only need to consider the first n points on our spatial grid. The Fourier transform assumes we are on a 2π periodic domain and so we have to transform our (L) domain into a (2π) periodic domain. We do this by rescaling the wavenumbers. We then apply the Fourier transform on these cubes and compute the average. We also remember to shift before we take the average. Then, we plot an isosurface to visualize the transformation.

We can see that there is one blob that is clearly bigger than the other blobs. This indicates that the particularly large blob must be corresponding to the central frequency. Hence, we find the maximum value in our averaged spectrum and the corresponding indices. These indices are then plugged into our grid of 2π periodic domain to get the center frequency. After obtaining the center frequency, we filter the unaveraged signal around this center frequency. We use the 3-D analog of the 1-D Gaussian filter, the specifics of which is discussed in the Theoretical Background section. To do this, we repeat taking the Fourier transform of each of the cubes. Then, we filter each signal around the center frequency and use the inverse Fourier transform to get data in the time domain. We have to remember to use the unshifted signal before we inverse transform it. For each of the cubes we find the maximum value and its corresponding index. Finally, we use that index to find the position of the marble in 3D. This process gives us the position of the marble in the dog's stomach for each measurement which helps us to plot its trajectory. This will be laid out in a ordered list later on.

2 Theoretical Background

2.1 Fourier Transform

We use the tool of Fourier transform to solve the problem of finding the marble. As we learned from our textbook [1], Fourier introduced the concept of representing a given function $f(x)$ by a trigonometric series of sines and cosines:

$$f(x) = \frac{a_0}{2} + \sum_{i=1}^{\infty} (a_n \cos nx + b_n \sin nx) \quad x \in (-\pi, \pi]. \tag{1}$$

The expansion in (1), by construction, produces 2π periodic functions since they are constructed from sines and cosines. This is the reason we rescale our wavenumbers. This idea leads to the Fourier transform we use. The Fourier transform is an integral transform defined over the entire line $x \in [-\infty, \infty]$ and has the following definition:

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

The inverse of the transform is defined as:

$$f(x) = \frac{1}{\sqrt{2\pi}} \int_{-\infty}^{\infty} e^{ikx} F(k) dk \tag{3}$$

2.2 Fast Fourier Transform

We specifically use the Fast Fourier Transorm developed by Cooley and Tukey. The transform is extremely efficient dropping the operation count for solving a system to $O(N \log N)$. This is a major difference from the operation count $O(N^3)$ for other standard algorithms like the Gaussian elimination and the LU decomposition. One of the key features of the transform is that it descretizes the range $[-L, L]$ into 2^n points. Also, since the integration Kernel $\exp(ikx)$ is oscillatory, the solutions on the interval $[-L, L]$ will have periodic boundary conditions. One big advantage of using the FFT is that it is extremely accurate in comparison with other standard discretization techniques. Thus, we use the Fast Fourier Transform in our problem of finding the marble.

2.3 Filtering using Gaussian Filter

However, the data we have is noisy due to certain movements by the dog and its internal body movements. The signal must be processed and filtered in order to determine the position of the marble. This is because we cannot be sure if the signal we received was a result of noise or the true signal itself. We treat the noise in the measurements as *white noise*. On top of that, the ultrasound produces more than one measurement. We have seen in class that white-noise can be modeled by adding a normally distributed random variable with zero mean and unit variance to each Fourier component of the spectrum[1]. The mean being zero beings that over many samples, the white-noise should add up to zero on average. Therefore, we average the shifted signal in the spectral domain using the number of measurements and expect to get a clean signal given that the number of measurements is not too small. It is important that we average in the frequency domain to produce a clear signature at the center frequency. In our case we have a time-varying signal, that is, the signal evolves over time. We can think of the movement of signal as the marble moving in time. Averaging in the time domain would get rid of this evolution and prevent us from obtaining the true signal. Once, we have averaged to find our center frequency we can use it in the process of spectral filtering.

Spectral filtering is a method which allows us to extract information at certain frequencies. In our case, we filter around the central frequency computed by averaging the spectrum. Filtering around this central frequency will removed undesired frequencies and the white noise associated with it. There are many different filters available to use. One of the simplest filters to consider is the Gaussian filter[1]

$$F(k) = e^{-\tau(k-k_0)^2)} \tag{4}$$

where τ measures the bandwidth of the filter and k is the wavenumber and k_0 is the central frequency. Application of the filter thins out frequencies away from k_0, If there is a signal around the center, the filter

helps us isolate the signal around this center. This is done by multiplying the signal in the spectral domain by the filter and then taking the inverse transform of this multiplied signal using (3).

For our particular case, we use the following 3D analog of the Gaussian filter:

$$\text{filter} = e^{-\tau((K_x-f_x)^2+(K_y-f_y)^2+(K_z-f_z)^2)} \tag{5}$$

where τ is still the bandwidth of the filter, K_x, K_y and K_z are 3D matrices of wavenumbers and f_x, f_y and f_z are scalars with the central frequency being $[f_x, f_y, f_z]$.

3 Algorithm Implementation and Development

In this section we lay out our steps to finding the trajectory of the marble in an orderly fashion. We also provide a pseudocode of the algorithm used in the next page. Steps:

1. Import data
2. Create the spatial domain grid from $-L$ to L
3. Compute the wavenumbers k and rescale them by $\frac{2\pi}{L}$
4. Create meshgrids $[X, Y, Z]$ and $[Kx, Ky, Kz]$ in the time and frequency domain respectively
5. Reshape the signal for each measurement
6. Compute the sum of the signals, shift it and take the average
7. Find the center frequency $[k_x, k_y, k_z]$ by finding the maximum frequency component in the averaged signal and its indices
8. Use the center frequency to create the filter: $e^{-\tau((Kx-k_x)^2+(Ky-k_y)^2+(Kz-k_z)^2))}$ where k_x, k_y, k_z are scalars and Kx, Ky, Kz are 3D matrices (take $\tau = 0.2$)
9. Take the Fourier transform of each measurement again and then multiply the shifted signal with the filter
10. Unshift the signal and take the inverse Fourier transform to get information about the position
11. Find the maximum value of each of the unshifted inverse Fourier transform and their corresponding indices (x_p, y_p, z_p)
12. Using the indices find the position by finding the corresponding co-ordinates in the meshgrid
13. Find the final position of the marble by taking it to be the last co-ordinate we find in the previous step

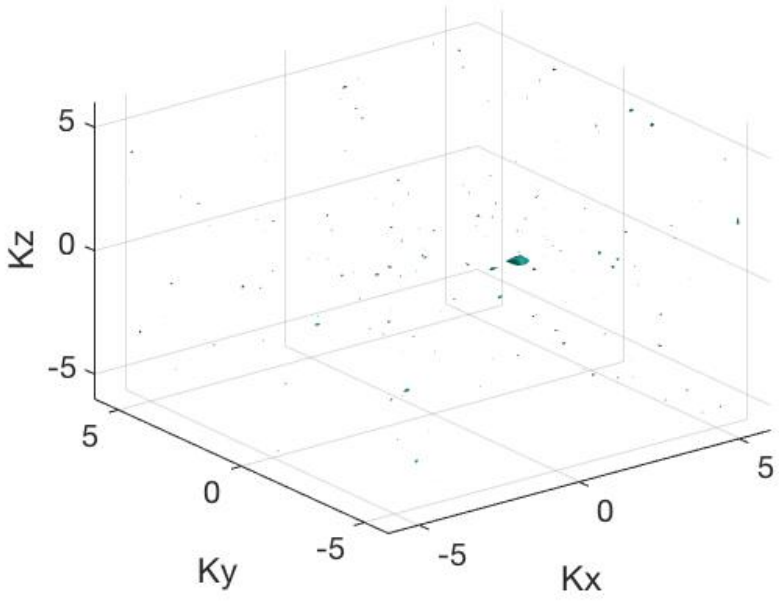

Figure 1: Isosurface of the averaged signal

4 Computational Results

The process of averaging the signal in the spectrum domain leads to the isosurface in Figure 1. We can notice that one blob is significantly larger than the others. This is our center frequency and we find it to be $[1.8850, -1.0472, 0]$. using the code provided in Appendix B. As mentioned above, we take this center frequency and use it to filter the signal in the original measurements. Figure 2 contains the isosurface of the inverse transform of the last measurement. We can see there is a clear blob in the picture and we just find the position of it. The process of finding the position in each measurement leads to the trajectory of the marble in Figure 3. The marble is spiraling downwards in a counterclockwise fashion. Finally, we find the final position of the marble to be $[-5.6250, 4.1218750, -6.0937750]$. This where the intense acoustic wave should be targeted to break up the marble and save Fluffy.

5 Summary and Conclusions

Using the tools of Fourier Analysis, we were successfully able to find the marble in the dog's stomach and save its life. Fourier analysis allowed us to denoise the data with the tools of averaging and filtering. This project sheds light on the usefulness of Fourier transform in a wide range of applications. This project also reflects how one particular tool might not be enough to solve the problem in hand and we have to combine different techniques and tools to arrive at a solution.

References

[1] Jose Nathan Kutz. *Data-driven modeling & scientific computation: methods for complex systems & big data*. Oxford University Press, 2013.

Appendix A MATLAB Functions

This section contains important MATLAB functions with a brief implementation explanation.

Algorithm 1: Averaging and Filtering

Import data from `Testdata.mat`
Create the spatial domain grid `x` and Fourier modes `k`
Rescale `k` by multiplying them by $\frac{2\pi}{L}$
Shift `k`
Create meshgrids `[X,Y,Z]` and `[Kx,Ky,Kz]`
Initialize `sum` =0
for $j = 1 : 20$ **do**
  Extract measurement j from `Undata` by reshaping into a 3D matrix
  Fourier transform the extracted measurement j
  sum $\leftarrow$ `sum`+ Fourier transform of j
end for
`sum` $\leftarrow$ shifted `sum`
`average` $\leftarrow \frac{\texttt{sum}}{20}$
Find maximum value of `average` and the corresponding indices
Convert linear indices to row and column subscripts
Find the values corresponding to the subscripts in `[Kx, Ky, Kz]`
`filter` $\leftarrow$ Gaussian filter for $\tau = 0.2$
Initialize position arrays $\texttt{X}_{\texttt{pos}}$, $\texttt{Y}_{\texttt{pos}}$, $\texttt{Z}_{\texttt{pos}}$ to the empty array
for $j = 1 : 20$ **do**
  Extract measurement j from `Undata` by reshaping into a 3D matrix
  Fourier transform the extracted measurement j
  Multiply the shifted transform with `filter`
  Unshift the filtered transform and take the inverse transform
  Find the maximum value of the inverse transform and the corresponding indices
  Convert the linear indices to row and column subscripts
  Find the values $\texttt{x}_\texttt{p}$, $\texttt{y}_\texttt{p}$, $\texttt{z}_\texttt{p}$ corresponding to the subscripts in `[X, Y, Z]`
  $\texttt{X}_{\texttt{pos}} \leftarrow [\texttt{X}_{\texttt{pos}}; \texttt{x}_\texttt{p}]$
  $\texttt{Y}_{\texttt{pos}} \leftarrow [\texttt{y}_{\texttt{pos}}; \texttt{Y}_\texttt{p}]$
  $\texttt{Z}_{\texttt{pos}} \leftarrow [\texttt{Z}_{\texttt{pos}}; \texttt{z}_\texttt{p}]$
end for

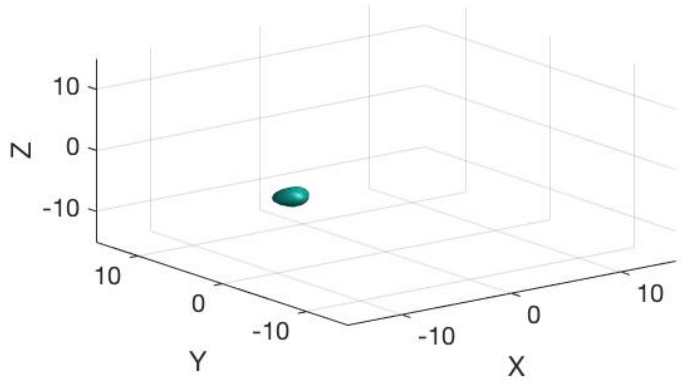

Figure 2: Isosurface of the inverse Fourier transform of the last measurement

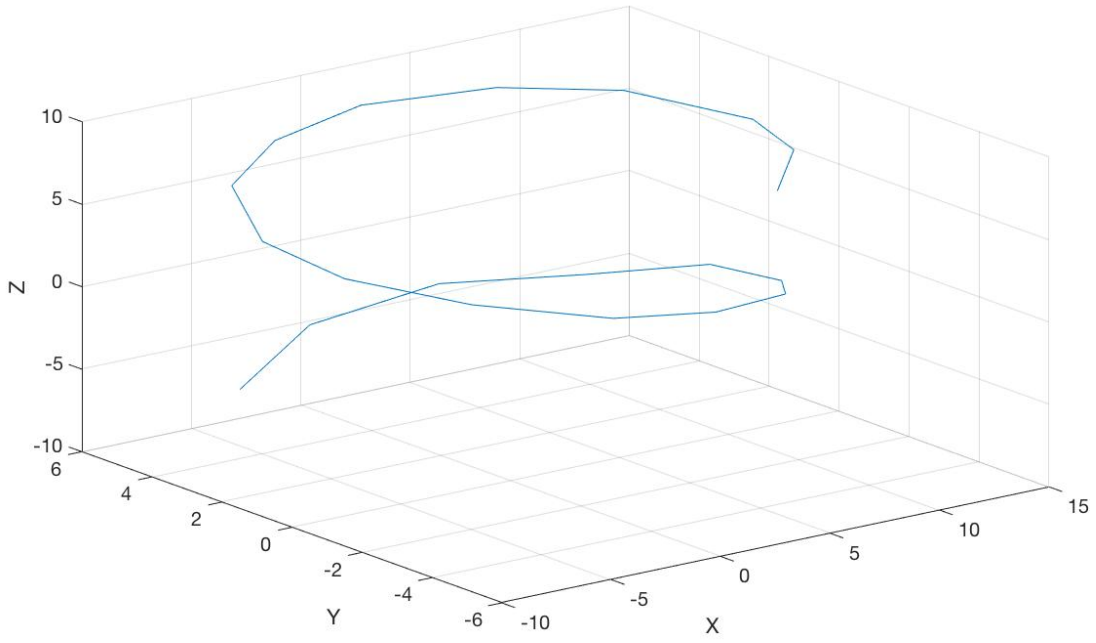

Figure 3: Trajectory of the marble in Fluffy's stomach

- `y = linspace(x1,x2,n)` returns a row vector of `n` evenly spaced points between `x1` and `x2`.
- `[X,Y] = meshgrid(x,y)` returns 2-D grid coordinates based on the coordinates contained in the vectors `x` and `y`. X is a matrix where each row is a copy of `x`, and `Y` is a matrix where each column is a copy of `y`. The grid represented by the coordinates `X` and `Y` has `length(y)` rows and `length(x)` columns.
- `Y = fftshift(X)` rearranges a Fourier transform `X` by shifting the zero frequency component to the center of the array.Since `X` is a multidimensional array in our case, this swaps half spaces of `X` along each dimension.
- `Y = fftn(x)` returns the multidimensional Fourier transform of an N dimensional array using a fast Fourier transform algorithm. This is equivalent to computing the 1-D transform `fft` along each dimension of X. The output Y is the same size as X. `ifftn` reverses this process.
- `[I1, I2, I3] = ind2sub(sz, index)` returns 3 arrays `I1, I2, I3` containing the three dimensional subscripts of the corresponding linear indices `index` for a 3D matrix of size `sz`, which is a vector with three elements specifying the size of each array dimension. `ifftshift` inverses this process.
- `B = reshape(A, sz)` reshapes `A` using the size vector `sz` which defines the size of `B`. In our case, `A` would be reshaped to a 64 x 64 x 64 array.
- `[m,i] = max(U)` returns the maximum `m` of the array `U` and the corresponding index `i` of the maximum value.

Appendix B MATLAB Code

```
clear all; close all; clc;
load Testdata
L=15; % spatial domain
```

```
n=64; % Fourier modes 2^6
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];%making it 2pi periodic
ks=fftshift(k); %shifting


%creating mesh grids
[X,Y,Z]=meshgrid(x,y,z);
[Kx,Ky,Kz]=meshgrid(ks,ks,ks);

%Part 1: Averaging to find the frequency signature

%initializing for averaging
uave = zeros;
for j =1:20
    uave = uave + fftn(reshape(Undata(j,:),n,n,n));  %adding the signal
end
utaves =fftshift(uave)/20;  %averaging the signal

figure(1)
close all, isosurface(Kx,Ky,Kz,abs(utaves)/max(abs(utaves(:))),0.6)
axis([-6 6 -6 6 -6 6]), grid on, drawnow
set(gca, 'FontSize', 20);
xlabel("Kx"); ylabel("Ky"); zlabel("Kz");


%finding the maximum and the corresponding index:
[m,index] = max(abs(utaves(:)));
[i_x,i_y,i_z] = ind2sub(size(abs(utaves)), index);

% finding the corresponding wavenumbers (frequency signature)
k_x = Kx(i_x,i_y,i_z);
k_y = Ky(i_x, i_y, i_z);
k_z = Kz(i_x,i_y,i_z);

freq_signature = [k_x, k_y, k_z]

%2. Filtering the data to denoise and find trajectory
tau =0.2;

% Using the center frequency to create the filter
filter = exp(-tau*((Kx - k_x).^2 + (Ky - k_y).^2 + (Kz - k_z).^2));

% Initializing vectors for the position
X_pos = [];
Y_pos = [];
Z_pos = [];
for j = 1:20
    un(:,:,:) = reshape(Undata(j,:),n,n,n);
    Ut = fftn(un);  % Fourier transform
    Uts = filter.*fftshift(Ut); % filtering
    Un = ifftn(ifftshift(Uts));  % Inverse transform
    [m,index] = max(abs(Un(:))); % Finding the max
    [x_p, y_p, z_p] = ind2sub(size(abs(Un)),index);
```

```
    X_pos = [X_pos; X(x_p,y_p, z_p)];
    Y_pos = [Y_pos; Y(x_p, y_p, z_p)];
    Z_pos = [Z_pos; Z(x_p, y_p, z_p)];
end

%Plotting isosurface of inverse Fourier transform of last measurement
figure(2)
close all, isosurface(X,Y,Z, abs(Un)/max(abs(Un(:))),0.6)
axis([-L L -L L -L L]), grid on, drawnow
set(gca, 'FontSize', 20);
xlabel("X"); ylabel("Y"); zlabel("Z");

%Plotting the trajectory
figure(3)
plot3(X_pos, Y_pos, Z_pos)
grid('on')
set(gca, 'FontSize', 20);
xlabel("X"); ylabel("Y"); zlabel("Z");


%Acoustic wave location(final position of marble):
position = [X_pos(20), Y_pos(20), Z_pos(20)];
```