AMATH 582 Homework 1

Ofir Kedar

January 27, 2021

Abstract

The purpose of this project was to use noisy acoustic data of an unknown central frequency emitted from a submarine, to determine the trajectory of the submarine to a precise estimation of where it might be at a given time stamp. The methods used to determine this were chosen as an introduction to signal processing. To find the central frequency emitted we implemented an averaging approach in frequency space. We filtered around this central frequency with a Gaussian filter to clean the data, and was then able extract precise estimations of where the submarine was at each discrete time stamp.

1 Introduction and Overview

When faced with an incoming signal emitted out of a single moving or static point in 3D space, we are able to locate the location of the source over time. This can be done through the averaging of the incoming signal in a time independent manner along with the filtering of that average central frequency. This concept is essential in systems like sonar technologies, as it is the fundamental process used to locate things such as enemy submarines. This is the problem we have posed with. Given noisy signal data about a submarine, we are asked to determine the frequency signature (central frequency). This is where we must time independently average our signal. We then are asked to plot a precise approximation of location of the of the submarine though out the time frame that the data is recorded. Here we filter our data of noise to get a clear precise reading. Lastly we are asked to extract the numerical location of the submarine at each time stamp of the data, otherwise referred to as the time realizations. To compute these answers, we will choose to use the Fourier Transform and Gaussian Filter.

2 Theoretical Background

Add your theoretical background here. Some example text: 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:

2.1 Fourier Transform

The Fourier transform, represented by equation 1, projects (or transforms) the spacial, or time domain, to the frequency domain. In the process of integrating over all of space-time, we then lose any information about time. The inverse transform is represented in equation 2.

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

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

2.2 Averaging

After transforming out data into the frequency domain, as noted in the previous subsection, information about time is suppressed from our data, and we can then average our frequency signals. This allows white noise to cancel out, and we can then take the max value in our matrix to be the location of out central frequency among all realizations (discrete measurements over a single time period) of our data.

2.3 Filtering

Once a central frequency is known, white noise can be eliminated by filtering out data with a 3D Gaussian filter around the central frequency in each direction. This is done by multiplying the frequency data at each realization by the Gaussian, represented for 1D filtering in equation 3, and 3D filtering in equation 4.

$$\text{1D Gaussian Filter} : f(k)e^{-\tau(k-k_0)2} \tag{3}$$

$$\text{3D Gaussian Filter} : f(k)(e^{-\tau((kx-k_0x)^2+(ky-k_0y)^2+(kz-k_0y)^2)}) \tag{4}$$

Here, $\boldsymbol{\tau}$ determines the width of the Gaussian (taken to equal 0.2 for the entirety of this experiment) and $\boldsymbol{k_0}$ being equal to the center frequency. by filtering the data, in frequency space for each realization, we can then invert the transformation, and show where the frequency is detected in 3D space, in each realization.

3 Algorithm Implementation and Development

3.1 Fourier Transform

While mathematically represented as an integral, the Fourier Transform can discreetly be represented as the the summations shown below and therefore be carried out algorithmicaly with discrete data.

$$\text{Descrete Fourier Transform} : x_k = \frac{1}{N}\sum_{n=0}^{N-1} x_n e^{(2\pi ikn)/N} \tag{5}$$

Where **N** is the number of Elements in your vector. While this is not explored to much depth in the Amath482 course, Matlab uses a function called fft, which stands for Fast Fourier Transform, which algorithmicaly carries out the transform using a divide and concur method on the DFT. While fft does this in a single dimension, fftn carries out in any n dimensions, and returns the decomposed frequencies that are associated with each dimension. This is what we use in this experiment to find the central frequencies in the Kx, Ky, and Kz directions.

3.2 Signal Data Extraction

The only algorithmic code provided with the assignment takes our data that comes in a 2D matrix containing 49 columns, and uses each column to create 49 64x64x64 matrices. Each matrix is a 3D representations of the signals in space at a one of the 49 discrete segments in time we are dealing with (the 49 realizations). Each matrix is then shown graphically, and the close all command is taken out it plots the matrices on top of each other, revealing the general path of the submarine through space.

Algorithm 1: Data Extraction

```
load subdata.mat
for j = 1 : 49 do
    Un(:, :, :)=reshape(subdata(:,j),n,n,n);
    M = max(abs(Un),[],'all');
    isosurface(X,Y,Z,abs(Un)/M,0.7)
    axis([-20 20 -20 20 -20 20])
    grid on
    drawnow
end for
```

3.3 Averaging

While creating each realization it is possible to breakdown the signal frequencies by simply applying the Fourier Transform to each realization with respect to new frequency axes in the frequency domain where we rescale our axes by a factor of 2/L to accommodate the transforms 2π periodic signal assumption. Once a realization is transformed, it is entirely time independent. This means that the we are looking at the frequencies alone, and their 3D location over time does not have any affect of on the data. The reason that this is useful is that now every realization matrix can be summed together then divided by 49 to result in an average at each element of the matrix, exposing the average frequencies. Now all existing frequencies can coexist in a singe matrix, where white noise is canceled out by itself as it fluctuates consistently in random with a mean value of approximately 0. By extracting the maximum value and its location in the average matrix, we now have the central frequency and its location in the frequency domain corresponding to each of our 3 dimensions.

Algorithm 2: Realization Averaging

```
load subdata.mat
for j = 1 : 49 do
   Un(:,:,:)=reshape(subdata(:,j),n,n,n);
   Unt = fftn(Un);
   ave = ave + Unt;
   M = max(abs(Unt),[],'all');
   isosurface(X,Y,Z,abs(Unt)/M,0.7)
   axis([-20 20 -20 20 -20 20])
   grid on
   drawnow
end for
ave = ave/49;
ave = fftshift(abs(ave)); % shift matrix to accomidate shift
aveMax = max(abs(ave),[],'all');
[Max_ave, Max_index] = max(abs(ave(:)));
[max_kx, max_ky, max_kz] = ind2sub([n,n,n], Max_index);
% Max_index gives index number of Max value in matrix ave
% max_k x gives index location of on the k x at Max index (same me for ky and kz)
```

3.4 Filtering

Now that, with the help of averaging in section 3.3, we have found our central frequency locations, we can use this information to filter the noise around our central frequency in each direction and pin point its discrete location at each realization. We do this using the aforementioned 3D Gaussian Filter from section 2.3. By multiplying each of the realizations by the filter in frequency space, all frequencies around the central frequency cancel to 0 and the central frequency in each direction is all that's left. When we inverse transform the realization, we are now left with 3D coordinates of the the locations where the filtered frequency exists. From there all that's left to do is take the max value in the X Y and Z directions, and find their location along each direction. These locations are the precise locations of the Submarine along each axis at the time of the particular realization at had. Duplicate this process for each realization, and you will be able to see the 3D location of the submarine at each discrete time stamp of each realization.

Algorithm 3: Filtering

tau = 0.2;
k0x = Kx(max_kx, max_ky, max_kz); % Finds Kx value at Max index
k0y = Ky(max_kx, max_ky, max_kz); % Finds Ky value at Max index
k0z = Kz(max_kx, max_ky, max_kz); % Finds Kz value at Max index
filter = fftshift(exp(-tau*(Kx - k0x).^{2}). $* exp(-tau * (Ky - k0y).^2). * exp(-tau * (Kz - k0z).^2)$);
for $j = 1 : 49$ **do**
  Un(:,:,:)=reshape(subdata(:,j),n,n,n);
  Unt = fftn(Un);
  Untf = filter.*Unt;
  Unf = ifftn(Untf);
  Max_ave, Max_index= max(abs(Unf(:)));
  max_x, max_y, max_z= ind2sub([n,n,n], Max_index);
  location_x(j) = X(max_x, max_y, max_z);
  location_y(j) = Y(max_x, max_y, max_z);
  location_z(j) = Z(max_x, max_y, max_z);
end for

4 Computational Results

4.1 Central Frequencies

Computed through averaging shown in section 3.3, and extracted for use in algorithm 3 the central frequencies in each direction are determined. The numerical results are shown in table 1.

k0x (Hz)	k0y (Hz)	k0z (Hz)
5.340707511102648	-6.911503837897545	2.199114857512855

Table 1: Cental Frecuencies

4.2 Submarine Coordinates

After identifying the central frequencies and filtering around them, we extract the location of the submarine at each realization along the x, y, and z coordinates, which can be used to plot the trajectory of the submarine. See Figure 1 in the page below. in order to locate in real time the exact location of the submarine, a P-8 Poseidon subtracking aircraft could be sent to the modeled location of the submarine according to our acoustic data. Table 2 below, shows the location of the submarine at each realization along the X Y axes.

5 Summary and Conclusions

The Fourier Transform is a powerful tool. It can be used to separate data from its time dependency and extract information in the frequency domain in a way that it can not be extracted in the space time domain. Averaging cannot occur in the same way in the outside of the frequency domain because averaging a position would not give us any useful information, and simply creates insignificant data that does not give us any information where something can be located over time. This is why finding the central frequencies is necessary to do outside of time.

The Guassian Filter is another strong tool to hold in our computational arsenal. filtering around our Central frequencies in any order of dimension allows us to make precise estimations of what portions of a signal is actually relevant, and weeds out any information that we know not to be of importance.

Realization	X coordinate	Y coordinate
1	3.1250	0
2	3.1250	0.31250
3	3.1250	0.62500
4	3.1250	1.2500
5	3.1250	1.5625
6	3.1250	1.8750
7	3.1250	2.1875
8	3.1250	2.5000
9	3.1250	2.8125
10	2.8125	3.1250
11	2.8125	3.4375
12	2.5000	3.7500
13	2.1875	4.0625
14	1.8750	4.3750
15	1.8750	4.6875
16	1.5625	5
17	1.2500	5
18	0.62500	5.3125
19	0.31250	5.3125
20	0	5.6250
21	-0.62500	5.6250
22	-0.93750	5.9375
23	-1.2500	5.9375
24	-1.8750	5.9375
25	-2.1875	5.9375
26	-2.8125	5.9375
27	-3.1250	5.9375
28	-3.4375	5.9375
29	-4.0625	5.9375
30	-4.3750	5.9375
31	-4.6875	5.6250
32	-5.3125	5.6250
33	-5.6250	5.3125
34	-5.9375	5.3125
35	-5.9375	5
36	-6.2500	5
37	-6.5625	4.6875
38	-6.5625	4.3750
39	-6.8750	4.0625
40	-6.8750	3.7500
41	-6.8750	3.4375
42	-6.8750	3.4375
43	-6.8750	2.8125
44	-6.5625	2.5000
45	-6.2500	2.1875
46	-6.2500	1.8750
47	-5.9375	1.5625
48	-5.3125	1.2500
49	-5	0.93750

Table 2: XY Location of Submarine at each time realization

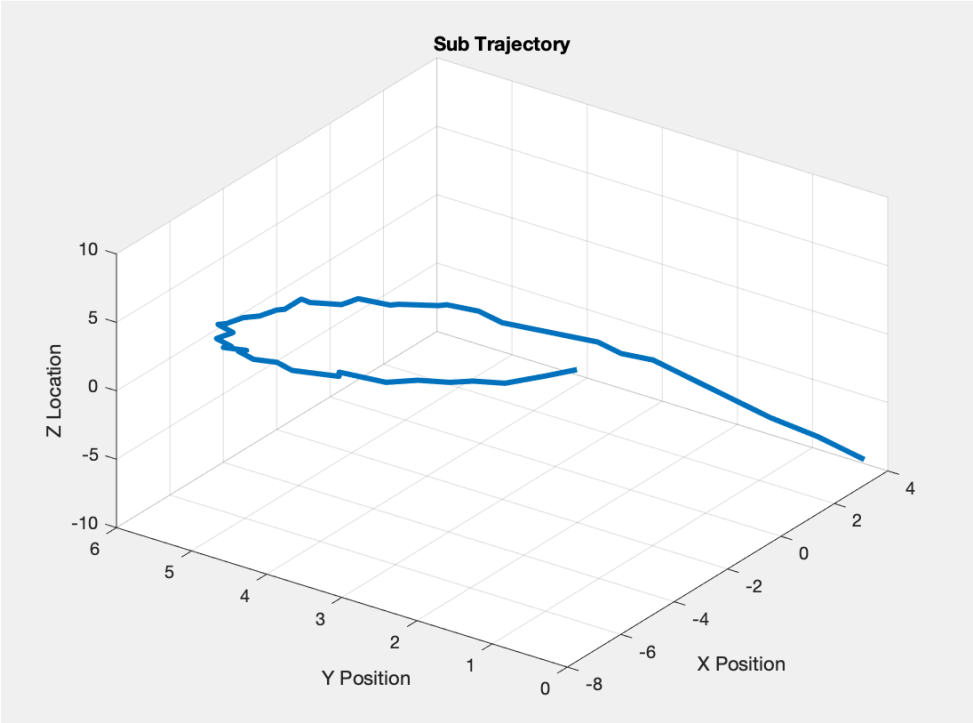

Figure 1: Submarine Coordinates Over Time Plot

References

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

Appendix A MATLAB Functions

Add your important MATLAB functions here with a brief implementation explanation. This is how to make an **unordered** list:

- `y = linspace(x1,x2,n)` returns a row vector of `n` evenly spaced points between `x1` and `x2`.
- `[X,Y,Z] = meshgrid(x,y,z)` returns 3-D grid coordinates defined by the vectors x, y, and z. The grid represented by X, Y, and Z has size length(y)-by-length(x)-by-length(z).
- Unt = fftn(Un) returns the Fourier Transformed matrix Unt of matrix Un in any number of dimensions that Un exists.
- Un = ifftn(Unt) returns the inverse Fourier Transformed Matrix Un of matrix Unt in any number of dimensions that Unt exists.
- [Max_ave, Max_index] = max(abs(ave(:))) returns the max absolute value in matrix ave and it's index location
- [I1,I2,I3,...,In] = ind2sub(SIZ,IND) returns N subscript arrays I1,I2,..,In containing the equivalent N-D array subscripts equivalent to IND for an array of size SIZ.
- fftshift(X) switches the left and right halves of the elements contained in X such that the 0 frequency component is centered along its axis

Appendix B MATLAB Code

```
% Clean workspace
clear all; close all; clc

%                   Data and Variables

load subdata.mat % Imports the data as the 262144x49 (space by time) matrix called subdata
L = 10; % spatial domain
n = 64; % Fourier modes
x2 = linspace(-L,L,n+1); %65 equaly spaced sections between -10 and 10
x = x2(1:n);
y = x;
z = x;
k = (2 * pi/(2 * L)) * [0:(n/2 - 1) -n/2:-1]; % assoasiates xy in 3d space to a frequency from 2pi
                                               % function to 2L function
ks = fftshift(k);
[X,Y,Z]=meshgrid(x,y,z);
[Kx,Ky,Kz] = meshgrid(ks,ks,ks); %frequency domain
ave = zeros(n,n,n); %added


%%
%                   Space-Time Signal Extraction

figure(1)
for j=1:49
    Un(:,:,:)=reshape(subdata(:,j),n,n,n);
    M = max(abs(Un),[],'all');
    isosurface(X,Y,Z,abs(Un)/M,0.7)
    axis([-20 20 -20 20 -20 20])
    grid on
    drawnow
end

%%
%                   Averaging

figure(2)

for j=1:49
    Un(:,:,:)=reshape(subdata(:,j),n,n,n);
    Unt = fftn(Un);
    ave = ave + Unt;
    M = max(abs(Unt),[],'all');
    isosurface(X,Y,Z,abs(Unt)/M,0.7)
    axis([-20 20 -20 20 -20 20])
    grid on
    drawnow
end
```

Listing 1: First half of code.

```
ave = ave/49;
ave = fftshift(abs(ave)); % shift matrix to accomidate shift
aveMax = max(abs(ave),[],'all');

figure(3)
isosurface(Kx,Ky,Kz,abs(ave)/aveMax,0.7)
axis([-20 20 -20 20 -20 20])
grid on
drawnow

[Max_ave, Max_index] = max(abs(ave(:))); % Max_index gives index number of Max value in matrix ave
[max_kx, max_ky, max_kz] = ind2sub([n,n,n], Max_index);  % max_kx gives index location of on the kx
                                                         % at Max index (same me for ky and kz)

%%
%                   Filtering

tau = 0.2;
k0x = Kx(max_kx, max_ky, max_kz);  % Finds Kx value at Max index
k0y = Ky(max_kx, max_ky, max_kz);  % Finds Ky value at Max index
k0z = Kz(max_kx, max_ky, max_kz);  % Finds Kz value at Max index
filter = fftshift(exp(-tau*(Kx - k0x).^2) .* exp(-tau*(Ky - k0y).^2) .* exp(-tau*(Kz - k0z).^2));

figure(4)
for j=1:49
    Un(:,:,:)=reshape(subdata(:,j),n,n,n);
    Unt = fftn(Un);
    Untf = filter.*Unt;
    Unf = ifftn(Untf);
    [Max_ave, Max_index] = max(abs(Unf(:)));
    [max_x, max_y, max_z] = ind2sub([n,n,n], Max_index);
    location_x(j) = X(max_x, max_y, max_z);
    location_y(j) = Y(max_x, max_y, max_z);
    location_z(j) = Z(max_x, max_y, max_z);
end

figure(4)
plot3(location_x, location_y, location_z, 'Linewidth', 3);
title('Sub Trajectory');
xlabel('X Position');
ylabel('Y Position');
zlabel('Z Location');
XY_coordinates = [location_x.' , location_y.'];
grid on
```

Listing 2: Second half of code.