AMATH 582 Homework 1

Clayton Heath

January 27, 2021

Abstract

The Fourier Transform has applications in signal processing. We will be using the Fourier Transform to analyze noisy acoustic data collected from a broad spectrum recording to hunt and track a submarine's location in the Puget Sound.

1 Introduction and Overview

We are given recording data of acoustics from some area in the Puget Sound. We have data measurements over a 24-hour period with half-hour increments in time, thus giving us 49 different measurements equally spaced throughout 24 hours. But, a lot of stuff is happening other than the submarine that we are trying to track, thus this data contains lots of noise. So, out of all the noise we will need to parse out the acoustic frequency given off by this submarine, then look at just that frequency for each time step in order to figure out its location throughout the 24-hour time period.

2 Theoretical Background

Given a function $f(x)$ for $x \in \mathbb{R}$, we define the *Fourier transform* of $f(x)$, written $\hat{f}(k)$ by the formula:

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

And if we are given $\hat{f}(x)$ and want to recover $f(x)$, we use the *inverse Fourier transform*:

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

Since we know e^{ikx} acts like $\sin(kx)$ and $\cos(kx)$, k represents the frequencies of these sine and cosine waves. Thus we have the the *Fourier transform* taking a function of x, where x is space or time, and converting it to a function of k, where k is frequency. But we see that for this method will require an infinite domain, which is not practical for physical situations. Thus we have the Fourier series:

$$f(x) = \sum_{-\infty}^{\infty} c_k e^{ik\pi x/L}, x \in [-L, L] \tag{3}$$

where

$$c_k = \frac{1}{2L} \int_{-L}^{L} f(x) e^{-ik\pi x/L} dx, k \in \mathbb{Z}$$

we see $\frac{\pi}{L}$ inside of the exponent of e, this is to force our function to be periodic over $[-\pi, \pi]$. We do this because the Fourier transform requires our function to be periodic in this way since sine and cosine functions are periodic in this way.

Now, we can derive the *discrete Fourier transform* (DFT), which is the Fourier series expect truncated at some maximum frequency. Given some vector of N values that are a function sampled at equally-spaced points, $\{x_0, x_1, x_2, ..., x_{N-1}\}$, then the DST is given by:

$$\hat{x}_k = \frac{1}{N} \sum_{n=0}^{N-1} x_n e^{\frac{2\pi ikn}{N}} \tag{4}$$

But we only have $k = 0, 1, ..., N-1$ frequencies present.
We also use a *filter function* to remove noise from a signal. The function chosen to use as a filter is the Gaussian in 3-dimensions:

$$g(x, y, z) = e^{-\tau\left((x-x_0)^2+(y-y_0)^2+(z-z_0)^2\right)} \tag{5}$$

Where x_0, y_0, z_0 are the frequency that we wish to filter around and τ is a positive constant. We use this as a filter since the exponential function will drop to near zero quickly as it leaves the coordinates it is centered around. Thus applying this filter in frequency space, it will essentially remove all frequencies not near our x_0, y_0, z_0. We can change τ to change how far away we get from our center before the function quickly drops to zero.

3 Algorithm Implementation and Development

First, we will look at the variables setup at the beginning of the MATLAB code. We have `L=10`, which as explained earlier, means $x, y, z \in [-10, 10]$. We also have `n=64` which are the Fourier nodes. Which also matches our $64 \times 64 \times 64$ matrices for each time. And then we have `k` which sets up our frequency domain.

Our first goal is to determine the signature frequency (center frequency) of the submarine. We do this through averaging of the spectrum. Under the `Averaging` section of the MATLAB code in Appendix B, we create the `ave` matrix to store our average. We then loop over all 49 instances, apply the fast Fourier transform, and sum them all together. We then take the absolute value of the sum and divide it by 49 (the number of recordings) to get the average. Why we do this is because, assuming the noise in the data has some distribution with mean 0, if we take the average of the frequencies at all times, the frequencies produced by the noise should reduce to near zero (by the law of large numbers). Thus, whatever frequency that has the largest value should be a submarine, since it must have been producing a consistent frequency across all of the 49 measurements. We can see the graph of these remaining frequencies in Figure 1. So, we then take the frequency of the maximum value of the average to be the signature frequency of the submarine.

Now that we know the center frequency, we can filter out all of the other frequencies so that we are able to track our submarine. Looking at the `Filtering` section of the MATLAB code, we use the Gaussian filter as shown in Equation 5, with the x_0, y_0, z_0, being the coordinates representing the signature frequency of the submarine. We arbitrarily let `tau=1`, we can change this a little bit to get very similar results, but too large or too small values of `tau` will cause wonky results since it will cause our Gaussian to fall to zero too quickly or two slowly. This would either not leave enough frequencies or leave too many frequencies. We then apply `fttshift()` to the Gaussian since we will be applying the Gaussian in the frequency domain. We then build a 3×49 matrix to prepare to store the x,y,z coordinates of the submarine at all 49 times.

We then again loop over all 49 measurements, for each one, we apply `fftn()`, then apply the filter using element-wise multiplication, causing all frequencies not close to our center to get close to 0. We then use `ifftn()` to bring our data back to the space domain, but now we are left only with the submarine and the noise removed. Thus we take the indices of our newly filtered data, to get the x, y, z coordinates of the submarine at that time using the `ind2sub()` function. We then get Table 1 which contains the coordinates of the submarine throughout the 24-hour period. We can see a plot of the path of the submarine in Figure 2.

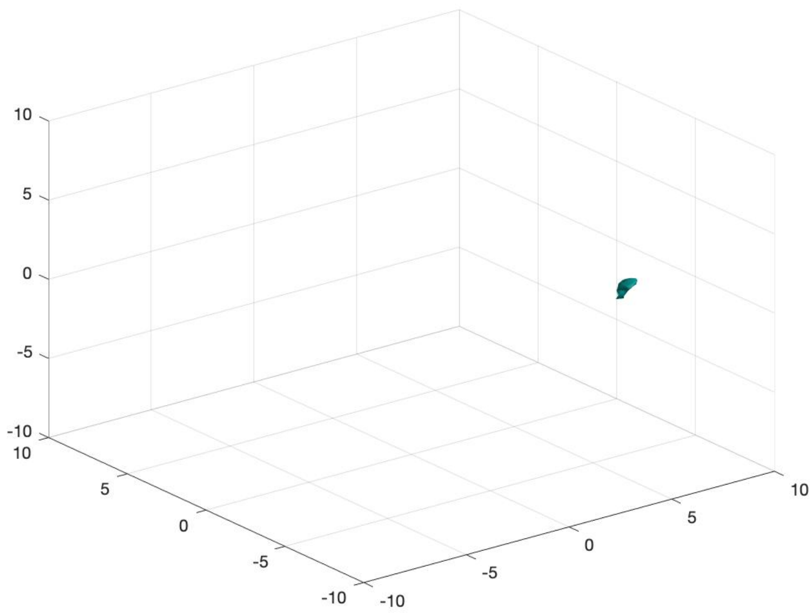

Figure 1: Here is the isosurface of our averaged data with an iso-value of 0.7.

4 Computational Results

For our center frequency, we obtained $x = 5.3407, y = -6.9115, z = 2.1991$.
See Table 1 for the coordinates of the submarine over the 24-period.
See Figure 1 for the isosurface of the averaged data from the `Averaging` section of the MATLAB code. This surface shows us the frequencies the remaining frequencies after cutting out what we assume to be noise.
See Figure 2 for the path of the submarine over the 24-hour period.

5 Summary and Conclusions

Given our noisy data of acoustics from some area in the Puget Sound, we were able to average the spectrum of our Fourier transformed data in order to find the center frequency generated by the submarine. We then took that center frequency and built ourselves at Gaussian filter centered at the frequency in order to de-noise the data. We do this by applying the filter to each instance of the transformed data, then transforming back to space, in order to retrieve the coordinates of the submarine at that time. We did this for all 49 instances and thus have the path of the submarine across the 24-hour period, and can have our aircraft follow the x, y coordinates from Table 1.

Appendix A MATLAB Functions

- `fftn(A)` returns the fast Fourier transform of `A` for the `n`-dimension.
- `ifftn(B)` returns the inverse fast Fourier transform of `B` for the `n`-dimension.
- `fftshift(X)` returns the values in the order: $\left\{\hat{x}_{-\frac{N}{2}}, \hat{x}_{-\frac{N}{2}+1}, ..., \hat{x}_{-1}, \hat{x}_0, \hat{x}_1, ..., \hat{x}_{\frac{N}{2}-1}\right\}$ given that $X = \left\{\hat{x}_0, \hat{x}_1, ..., \hat{x}_{\frac{N}{2}-1}, \hat{x}_{\frac{N}{2}}, \hat{x}_{-\frac{N}{2}+1}, ..., \hat{x}_{-1}\right\}$. Also applies for higher dimensions.

Time	X	Y	Z
0	3.1416	0	-8.1681
1	3.1416	0.3142	-7.8540
2	3.1416	0.6283	-7.5398
3	3.1416	1.2566	-7.2257
4	3.1416	1.5708	-6.9115
5	3.1416	1.8850	-6.5973
6	3.1416	2.1991	-6.2832
7	3.1416	2.5133	-5.9690
8	3.1416	2.8274	-5.6549
9	2.8274	3.1416	-5.3407
10	2.8274	3.4558	-5.0265
11	2.5133	3.7699	-4.7124
12	2.1991	4.0841	-4.3982
13	1.8850	4.3982	-4.0841
14	1.8850	4.7124	-3.7699
15	1.5708	4.7124	-3.4558
16	1.2566	5.0265	-3.1416
17	0.9425	5.3407	-3.1416
18	0.3142	5.3406	-2.5133
19	0.0	5.6540	-2.1991
20	-0.3142	5.6549	1.8850
21	-0.9425	5.9690	-1.8850
22	-1.2566	5.9690	-1.2566
23	-1.8850	5.9690	-1.2566
24	-2.1991	5.9690	-0.9425
25	-2.8274	5.9690	-0.6283
26	-3.1416	5.9690	-0.3142
27	-3.7699	5.9690	0
28	-4.0841	5.9690	0.3142
29	-4.3982	5.9690	0.6283
30	-4.7124	5.6549	0.9425
31	-5.3407	5.6549	1.2566
32	-5.6549	5.3407	1.5708
33	-5.9690	5.3407	1.8850
34	-5.9690	5.0265	2.1991
35	-6.2831	4.7124	2.5133
36	-6.5973	4.7124	2.8274
37	-6.9115	4.3982	3.1416
38	-6.9115	4.0841	3.4558
39	-6.9115	4.0841	3.7699
40	-6.9115	3.4558	4.0841
41	-6.9115	3.4558	4.3982
42	-6.9115	3.1416	4.7124
43	-6.5973	2.5133	5.0265
44	-6.5973	2.1991	5.0265
45	-6.2832	1.8850	5.6549
46	-5.6549	1.5708	5.5649
47	-5.6549	1.2566	6.2832
48	-5.0265	0.9425	6.5973

Table 1: The X, Y, and Z coordinates of the submarine where each time represents 30 minutes passing.

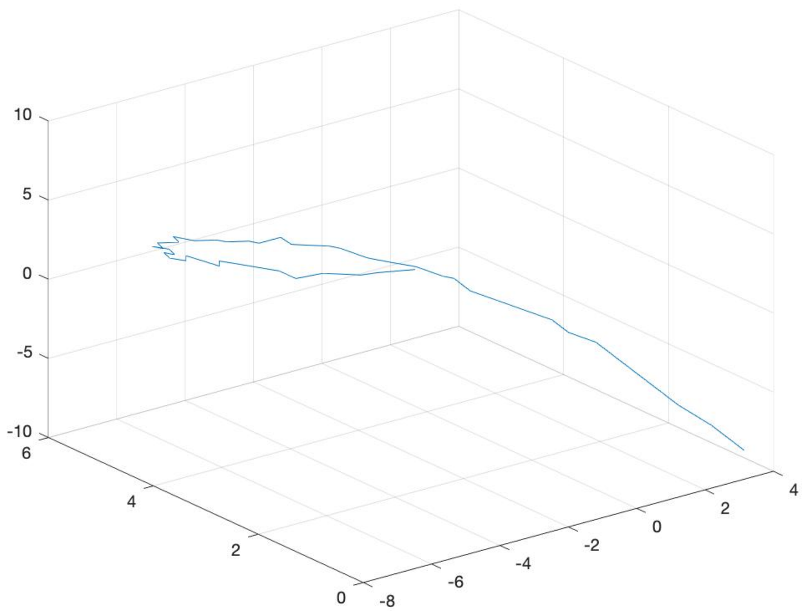

Figure 2: Here is the path of the submarine over the 24-hour period.

- `[I1,I2,...,In] = ind2sub(sz,ind)` returns n arrays I1,I2,...,In containing the equivalent multidimensional subscripts corresponding to the linear indices `ind` for a multidimensional array of size `sz`.

Appendix B MATLAB Code

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

load subdata.mat % Imports the data as the 262144x49 (space by time) matrix called subdata 5
L = 10; % 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);
%% Averaging
ave = zeros(n,n,n);

for j=1:49 % Averages all 49 instances of Fourier Transformation
    Un(:,:,:)=reshape(subdata(:,j),n,n,n);
    utn = fftn(Un);
    ave = ave + utn;
end
ave = abs(fftshift(ave))/49;
[M, ind] = max(abs(ave),[],'all','linear');
close all, isosurface(Kx,Ky,Kz,abs(ave)/M,0.7)
axis([-10 10 -10 10 -10 10]); grid on, drawnow
[x0, y0, z0] = ind2sub([64 64 64], ind);
% Finds center frequency
center_Kx = Kx(x0,y0,z0);
center_Ky = Ky(x0,y0,z0);
center_Kz = Kz(x0,y0,z0);
%% Filtering
tau = 1;
% Gaussian Filter
filter = fftshift(exp(tau*(-(Kx-center_Kx).^2 - (Ky-center_Ky).^2)-(Kz-center_Kz).^2));
pxyz = zeros(3,49);
% Applies the Gaussian filter and finds the coordinates of the sub for each time
for j=1:49
    Un(:,:,:) = reshape(subdata(:,j),n,n,n);
    utn = ifftn(fftn(Un).*filter);
    [M, ind] = max(abs(utn),[],'all', 'linear');
    [xc, yc, zc] = ind2sub([64 64 64], ind);
    pxyz(1,j) = Kx(xc,yc,zc);
    pxyz(2,j) = Ky(xc,yc,zc);
    pxyz(3,j) = Kz(xc,yc,zc);
end
plot3(pxyz(1,:),pxyz(2,:),pxyz(3,:))
grid on
```